

**THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Applicants: Gero Offer
Appl. No.: 09/899,435
Conf. No.: 3369
Filed: July 5, 2001
Title: TELECOMMUNICATION NETWORK, METHOD OF OPERATING SAME,
AND TERMINAL APPARATUS THEREIN
Art Unit: 2153
Examiner: Philip J. Chea
Docket No.: 0112740-257

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' APPEAL BRIEF

Sir:

Appellants submit this Appeal Brief in support of the Notice of Appeal filed on May 22, 2006. This Appeal is taken from the Final Rejection in the Office Action dated February 22, 2006.

I. REAL PARTY IN INTEREST

The real party in interest for the above-identified patent application on appeal is Siemens Aktiengesellschaft, by virtue of an Assignment dated September 12, 2001 and recorded at the United States Patent and Trademark Office at reel 012241, frame 0293.

II. RELATED APPEALS AND INTERFERENCES

Appellants' legal representative and the Assignee of the above-identified patent application do not know of any prior or pending appeals, interferences or judicial proceedings which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision with respect to the above-identified Appeal.

III. STATUS OF CLAIMS

Claims 1-25 are pending in the above-identified patent application. Claims 1-25 stand rejected. Therefore, Claims 1-25 are being appealed in this Brief. A copy of the appealed claims is included in the Claims Appendix.

IV. STATUS OF AMENDMENTS

A Final Office Action was mailed on February 22, 2006. Appellants filed a Pre-Appeal Brief Conference Request on May 22, 2006. After reviewing the rejection, the pre-appeal conference could not agree to a resolution and forwarded the application to this Board via the decision dated August 1, 2006. A copy of the Final Office Action is attached as **Exhibit A** in the Evidence Appendix.

V. SUMMARY OF CLAIMED SUBJECT MATTER

A summary of the invention by way of reference to specification and/or figures for each of the independent claims is provided as follows:

Independent Claim 1 is directed to a telecommunication network that includes a central server (S) of an access or service provider, where the central server includes an interrogation part (3) for interrogating hardware and software configurations of a plurality of terminal devices ([0025]). The server also includes a software transmitting part [0012, 0017] for loading software and/or data that is customized according to the detected hardware and software configuration one of the plurality of terminal devices [0029]. A plurality of terminal devices include a response transmitting part for transmitting a configuration code [0026, 0032] identifying the respective hardware and software configuration to the central server in response to an inquiry by the interrogation part. Each of the plurality of terminal devices also including a software receiving part for receiving and internally storing at least one of the transferred software and data, where the interrogation part and the response transmitting part interrogate the respective hardware and software configuration and to transmit the respective configuration code when the terminal device logs onto the telecommunication network, or when predetermined times or time intervals occur [0011, 0033; FIG. 1]. Distributed control parts, which are distributed in both the central server and the plurality of terminal devices, implement an interactive control over the software transmitting part, and for the interactive specifying of a charging mode for at least one of downloaded software and downloaded data [0017, 0020, 0026-27].

Independent Claim 21 is directed to terminal device for use in a telecommunication network having a plurality of terminal devices of users, each having a predetermined hardware and software configuration, and a central server of an accessor service provider, where the terminal device includes a response transmitting part for transmitting a configuration code that identifies a currently implemented hardware and software configuration to the central server in response to an interrogation by the central server [0025-26, 0029, 0032]. Distributed control parts implement an interactive control of the downloading of software and/or data from the central server [0017, 0035]. A charging mode display part displays at least one available charging mode for one of offered and selected software and data, where a charging mode confirmation part specifies the charging mode [0031-32].

Although specification citations are given in accordance with C.F.R. 1.192(c), these reference numerals and citations are merely examples of where support may be found in the specification for the terms used in this section of the Brief. There is no intention to suggest in any way that the terms of the claims are limited to the examples in the specification. As demonstrated by the reference numerals and citations, the claims are fully supported by the specification as required by law. However, it is improper under the law to read limitations from the specification into the claims. Pointing out specification support for the claim terminology as is done here to comply with rule 1.192(c) does not in any way limit the scope of the claims to those examples from which they find support. Nor does this exercise provide a mechanism for circumventing the law precluding reading limitations into the claims from the specification. In short, the reference numerals and specification citations are not to be construed as claim limitations or in any way used to limit the scope of the claims.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-4, 14, 21, 22 and 24 were rejected under 35 U.S.C. 103(a) as being unpatentable over *Kobata* (U.S. Patent No. 4,540,585) in view of *Nakagawa* (U.S. Patent 5,835,911);

Claims 5-6 were rejected under 35 U.S.C. 103(a) as being unpatentable over *Kobata* (U.S. Patent No. 4,540,585) in view of *Nakagawa* (U.S. Patent 5,835,911), and further in view of *Chen et al.* (U.S. 5,797,016);

Claims 7, 12, 13, and 23 were rejected under 35 U.S.C. 103(a) as being unpatentable over *Kobata* (U.S. Patent No. 4,540,585) in view of *Nakagawa* (U.S. Patent 5,835,911), and further in view of *Valentine* (U.S. 6,018,654);

Claim 8 was rejected under 35 U.S.C. 103(a) as being unpatentable over *Kobata* (U.S. Patent No. 4,540,585) in view of *Nakagawa* (U.S. Patent 5,835,911), and further in view of *Pepe et al.* (U.S. Patent 5,742,668);

Claims 9-11 and 25 were rejected under 35 U.S.C. 103(a) as being unpatentable over *Kobata* (U.S. Patent No. 4,540,585) in view of *Nakagawa* (U.S. Patent 5,835,911), and further in view of *Shear* (U.S. Patent 4,977,594);

Claims 15 and 17 were rejected under 35 U.S.C. 103(a) as being unpatentable over *Kobata* (U.S. Patent No. 4,540,585) in view of *Shah* (U.S. Patent 6,029,065) and further in view of *Nakagawa* (U.S. Patent 5,835,911);

Claims 19 and 20 were rejected under 35 U.S.C. 103(a) as being unpatentable over *Kobata* (U.S. Patent No. 4,540,585) in view of *Shah* (U.S. Patent 6,029,065) in view of *Nakagawa* (U.S. Patent 5,835,011) and further in view of *Shear* (U.S. Patent 4,977,594). A copy of *Kobata*, *Nakagawa*, *Chen*, *Valentine*, *Pepe*, *Shear* and *Shah* are attached herewith as **Exhibits B, C, D, E, F, G and H**, respectively.

VII. ARGUMENT

A. LEGAL STANDARDS

1. Obviousness under 35 U.S.C. § 103

The Federal Circuit has held that the legal determination of an obviousness rejection under 35 U.S.C. § 103 is:

whether the claimed invention as a whole would have been obvious to a person of ordinary skill in the art at the time the invention was made...The foundational facts for the *prima facie* case of obviousness are: (1) the scope and content of the prior art; (2) the difference between the prior art and the claimed invention; and (3) the level of ordinary skill in the art...Moreover, objective indicia such as commercial success and long felt need are relevant to the determination of obviousness...Thus, each obviousness determination rests on its own facts.

In re Mayne, 41 U.S.P.Q. 2d 1451, 1453 (Fed. Cir. 1997).

In making this determination, the Patent Office has the initial burden of proving a *prima facie* case of obviousness. *In re Rijckaert*, 9 F.3d 1531, 1532, 28 U.S.P.Q. 2d 1955, 1956 (Fed. Cir. 1993). This burden may only be overcome “by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings.” *In re Fine*, 837 F.2d 1071, 1074, 5 U.S.P.Q. 2d 1596, 1598 (Fed. Cir. 1988). “If the examination at the initial stage does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of the patent.” *In re Oetiker*, 24 U.S.P.Q. 2d 1443, 1444 (Fed. Cir. 1992).

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the reference or references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. *In re Fine*, 837 F.2d 1071, 5, U.S.P.Q.2d 1596 (Fed. Cir. 1988). Second there must be a reasonable expectation of success. *In re Merck & Co., Inc.*, 800 F.2d 1091, 231 U.S.P.Q. 375 (Fed. Cir. 1986). Finally, all of the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q., 580 (CCPA 1974).

Further, the Federal Circuit has held that it is “impermissible to use the claimed invention as an instruction manual or ‘template’ to piece together the teachings of the prior art so that the

claimed invention is rendered obvious.” *In re Fritch*, 23 U.S.P.Q.2d 1780, 1784 (Fed. Cir. 1992). “One cannot use hindsight reconstruction to pick and choose among isolated disclosures in the prior art to deprecate the claimed invention” *In re Fine*, 837 F.2d 1071 (Fed. Cir. 1988).

Moreover, the Federal Circuit has held that “obvious to try” is not the proper standard under 35 U.S.C. §103. *Ex parte Goldgaber*, 41 U.S.P.Q.2d 1172, 1177 (Fed. Cir. 1996). “An-obvious-to-try situation exists when a general disclosure may pique the scientist curiosity, such that further investigation might be done as a result of the disclosure, but the disclosure itself does not contain a sufficient teaching of how to obtain the desired result, or that the claimed result would be obtained if certain directions were pursued.” *In re Eli Lilly and Co.*, 14 U.S.P.Q.2d 1741, 1743 (Fed. Cir. 1990).

Of course, references must be considered as a whole and those portions teaching against or away from the claimed invention must be considered. *Bausch & Lomb, Inc. v. Barnes-Hind/Hydrocurve Inc.*, 796 F.2d 443 (Fed. Cir. 1986). “A prior art reference may be considered to teach away when a person of ordinary skill, upon reading the reference would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the Applicant.” *Monarch Knitting Machinery Corp. v. Fukuhara Industrial Trading Co., Ltd.*, 139 F.3d 1009 (Fed. Cir. 1998), quoting, *In re Gurley*, 27 F.3d 551 (Fed. Cir. 1994).

B. THE CLAIMED INVENTION – CLAIM 1

Independent Claim 1 recites, in part, a central server having an interrogation part for interrogating hardware and software configurations of a plurality of terminal devices and a plurality of terminal devices including a response transmitting part for transmitting a configuration code identifying the respective hardware and software configuration to the central server in response to an inquiry by the interrogation part. The interrogation part and the response transmitting part interrogate the respective hardware and software configuration and transmit the respective configuration code when at least one of the terminal device logs onto the telecommunication network, predetermined times occur, and predetermined time intervals occur. Distributed control parts are distributed in both the central server and the plurality of terminal devices, where the distributed control parts implement an interactive control over the software

transmitting part, and being constructed for the interactive specifying of a charging mode for at least one of downloaded software and downloaded data.

C. THE REJECTION OF CLAIM 1 UNDER 35 U.S.C. §103(A) TO *KOBATA* AND *NAKAGAWA* SHOULD BE REVERSED BECAUSE THE PATENT OFFICE HAS NOT ESTABLISHED A *PRIMA FACIE* CASE OF OBVIOUSNESS

1. One having ordinary skill in the art would not be motivated to combine *Kobata* with *Nakagawa* in the manner suggested in the Final Office Action to arrive at the present claims

Appellants respectfully submit that the examiner has not taken into consideration the teachings *as a whole* in the cited prior art, and has further engaged in impermissible hindsight in formulating this rejection. As this Board is aware, the Federal Circuit has held that it is “impermissible to use the claimed invention as an instruction manual or ‘template’ to piece together the teachings of the prior art so that the claimed invention is rendered obvious.” *In re Fritch*, 23 U.S.P.Q.2d 1780, 1784 (Fed. Cir. 1992). Indeed, references must be considered as a whole and those portions teaching against or away from the claimed invention must be considered. *Bausch & Lomb, Inc. v. Barnes-Hind/Hydrocurve Inc.*, 796 F.2d 443 (Fed. Cir. 1986).

The entire disclosure of *Kobata* is directed to a “market delivery system,” which is essentially advertisement delivery software that delivers advertisements or other “targeted” data from a server to users based on information and demographics collected from those users (col. 4, lines 22-25). *Kobata* discloses a system where software is first provided to each user in the network, and each user must execute - and permit - the software to transmit information about their computers and/or their software to the provider 10 (col. 3, lines 52-60; col. 4, lines 43-56; see claim 1: “when the client system executes the software”). The provider then receives an aggregate of information regarding various users’ hardware and software physically residing on their computers (col. 4, line 57 – col. 5, line 5). Having this information, the provider can then tailor advertisements directed to users, and send advertisements that would be optimized by the software residing on a user’s computer (e.g., PDF vs. MPEG), as well as the hardware residing therein (e.g., sound cards, video cards) (see col. 5, lines 1-5, 18-21). The configuration in *Kobata* is clearly disclosed as a “push” system, where, once a user has subscribed to the advertising service and the information from the users is initially received, the advertisements will begin to flow to the users from the server without further interaction (col. 5, lines 6-18, 33-

37). Like most every “push” system, such transmissions are transparent to the user, and are intended to be used with minimal interaction (col. 4, lines 43-49).

Along a completely different line of software networking, *Nakagawa* teaches a system and method where an application program on a user’s computer detects when software subject to maintenance is activated and transmits an inquiry over the network to the software vendor’s computer for information on the current version of the software. The server program compares data in the inquiry with data relating to the latest version of the software and returns update instruction information and updated software if appropriate (see Abstract). Thus, while the software and hardware information concerning a user’s computer is initially sent to the software provider, the subsequent updates are premised only upon the software version residing on a user’s computer (col. 1, lines 13-19; col. 8, lines 26-36). *Nakagawa* further teaches an “access qualification level” for customers (col. 67, lines 50-60) where customers are offered various functionalities, based on their qualification level. Each qualification level assigned by the seller to the software allows users, depending on their qualification, to access different software at different levels of access (payment type, time of payment, etc. – col. 7, lines 39-48).

It is apparent to Applicant that there is no teaching suggestion or motivation for one having ordinary skill in the art to combine *Kobata* with *Nakagawa*. As discussed above, *Kobata* is a system in which advertisements are targeted to users that choose to volunteer their PC hardware and software information so that an optimal advertising campaign can be formulated for them by advertisers – no software is being “updated”. Furthermore, the “push” configuration makes it so that users transparently receive advertisements. Applicant submits *Kobata* teaches away from the presently claimed features, and further teaches away from *Nakagawa*. It is not understood why an interactive control would be implemented in *Kobata* to specify a “charging mode” mode for the downloaded advertisements. If the proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification. *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984). To suggest that users in *Kobata* would interactively pay the advertiser to receive advertisements misinterprets the teaching in the reference and of “push” systems in general, and runs contrary to common sense. Also, the access qualification in *Nakagawa* goes to upgrading and purchasing software on a networked user system so that buyers and sellers may effectively negotiate these transactions. In *Kobata*, there is no software to

“upgrade,” but only content that is provided to the users (col. 4, lines 22-25, 34-42). Similarly, it is not understood how incorporating the access levels of *Nakagawa*, which restricts user access to software, would be feasible in the advertising-model system of *Kobata*. Again, it wouldn’t make sense to have advertisers create restrictions to their own advertisements on putative consumers. Because of at least these differences, Appellants respectfully submit that the Examiner has failed to provide a sufficient basis or motivation for combining the cited references. Consequently, in view of the portions of the cited references teaching away each other and from the present claims, one having ordinary skill in the art would not be motivated to modify or combine the cited references to arrive at the present claims.

2. *Kobata* and *Nakagawa*, alone or in combination, fail to disclose or suggest all of the recited feature of claim 1

Appellants respectfully submit that, even if combinable, the cited references do not disclose or suggest all of the claimed elements. For example, claim 1 recites “a central server having an interrogation part for interrogating hardware and software configurations of a plurality of terminal devices” where the terminal devices have “a response transmitting part for transmitting a configuration code identifying the respective hardware and software configuration to the central server in response to an inquiry by the interrogation part.”

Kobata discloses a client software package (64) that is first installed at a client’s computer. The client then subsequently executes the software, whereupon the software transmits the client’s software/hardware configuration (col. 4, lines 57-67). Thus, the corresponding data transfer is initiated solely on the client’s side (col. 3, lines 6-9, 52-60), and cannot be considered a result of an “interrogation,” either in the technical or conventional sense. There is no “response” being initiated by the client. Moreover, the response is not in the form of a configuration code, as recited in the present claims.

Nakagawa discloses a client sending information about a software version in response to a query, however, *Nakagawa* does not disclose a software and hardware configuration being transmitted, and also does not disclose a configuration code.

Also, claim 1 recites that the interrogation part and the response transmitting part interrogate the hardware and software configuration and transmit the respective configuration code when at least one of the terminal device (1) logs onto the telecommunication network, (2) predetermined times occur, and (3) predetermined time intervals occur. *Kobata* clearly fails to teach this configuration.

Moreover, claim 1 recites distributed control parts, which are distributed in both the central server and the plurality of terminal devices, where the distributed control parts implement an interactive control over the software transmitting part, and specifying a charging mode for at least one of downloaded software and downloaded data. As discussed above, *Kobata* does not have any control parts distributed to implement interactive control for charging modes. Regarding *Nakagawa*, the reference teaches that the client's program directs a query to a software provider's server over a network (col. 1, lines 13-19; col. 8, lines 26-36). The "access qualification level" disclosed in *Nakagawa* only refers to control being implemented on the server side only (col. 67, lines 50-60). In other words, the access qualification levels in *Nakagawa* are a indicator of user rights that are defined by the seller on the server, as opposed to the server and the terminal devices – interactive control of the software transmission is controlled on the server side exclusively (see also col. 67, lines 39-48).

As shown above, the cited references fail to disclose or suggest unique aspects of the present claims. Though one may argue that each piece is an independent element, each element is necessary and makes this inventive aspect unique. Consequently, the above-cited references fail to disclose or suggest while teaching away from the above unique aspects of the present claims. As a result of the above discussion, Appellants have met the burden of proof to show that *Kobata* and *Nakagawa* fail to render obvious the present claims. Appellants respectfully submit that the Examiner has improperly applied hindsight reasoning by selectively piecing together teachings of each of the references in an attempt to recreate what the claimed invention discloses. As the Federal Circuit explained, one cannot use "hindsight reconstruction to pick and choose among isolated disclosures in the prior art" to re-create the claimed invention. *In re Fine*, 5 U.S.P.Q. 2d 1596 (Fed. Cir. 1988). Further, it is "impermissible to use the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious." *In re Fritch*, 23 U.S.P.Q.2d 1780, 1784 (Fed. Cir.

1992). Accordingly, Appellants respectfully submit that Claims 1-14 are novel, nonobvious and distinguishable from the cited references and are in condition for allowance.

D. THE REJECTION OF CLAIM 15 UNDER 35 U.S.C. §103(A) TO *KOBATA, SHAH* AND *NAKAGAWA* SHOULD BE REVERSED BECAUSE THE PATENT OFFICE HAS NOT ESTABLISHED A *PRIMA FACIE* CASE OF OBVIOUSNESS

1. One having ordinary skill in the art would not be motivated to combine *Kobata* with *Shah* and *Nakagawa* in the manner suggested in the Final Office Action to arrive at the present claims

For the same reasons recited above in connection with claim 1, there is no teaching, suggestion or motivation for one having ordinary skill in the art to combine these reference in the manner suggested in the Office Action. With regard to *Shah*, an interactive menu is provided for users that wish to accept or reject features available for a base station in a mobile telecommunication network (col. 9, lines 54-61). However, the Office Action doesn't reconcile how the signaling and network considerations for an interactive menu on the wireless network of *Shah* (i.e., paging channel/access channel) would be incorporated into the line-based system of *Kobata*. Under *Shah*, when a mobile user visits a network, registration procedures are used to enable the visited network to identify the mobile unit network connection and paging purposes, and the mobile station's home network, for billing purposes. Once registered, the base station will download information to the mobile station which will notify the mobile station of which network features are available and how they may be accessed in the local network (col. 3, lines 31-40). However none of the system requirements of *Shah* – i.e., home location registers, base stations, mobile registration, etc. – is remotely applicable to the teaching of *Kobata*. Furthermore, as in *Nakagawa*, it is not explained how or why an interactive menu would be needed for the “push” system of *Kobata*. Again, Appellant respectfully submits this rejection, as in claim 1, was formulated through the use of impermissible hindsight, and without considering each reference as a whole when applying it to the presently claimed features.

2. *Kobata, Shah and Nakagawa*, alone or in combination, fail to disclose or suggest all of the recited feature of claim 15

For the same reasons recited above in connection with claim 1, *Kobata, Shah* and *Nakagawa* fail to teach or suggest the recited features of claim 15. Accordingly, Appellants have met the burden of proof to show that *Kobata, Shah* and *Nakagawa* fail to render obvious the present claims. Appellants respectfully submit that the Examiner has also improperly applied hindsight reasoning by selectively piecing together teachings of each of the references in an attempt to recreate what the claimed invention discloses. As such, Appellants respectfully submit that Claims 15-25 are novel, nonobvious and distinguishable from the cited references and are in condition for allowance.

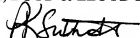
VIII. CONCLUSION

Appellants respectfully submit that the Patent Office has failed to establish a *prima facie* case of obviousness under 35 U.S.C. §103(a) with respect to the rejections of Claims 1-25. Accordingly, Appellants respectfully submit that the obviousness rejections are erroneous in law and in fact and should therefore be reversed by this Board. The Director is authorized to charge any additional fees which may be required, or to credit any overpayment to Deposit Account No. 02-1818. If such a withdrawal is made, please indicate the Attorney Docket No. 112740-257 on the account statement.

Respectfully submitted,

BELL, BOYD & LLOYD LLC

BY



Patricia Kane Schmidt
Reg. No. 46,446
Customer No. 29177

Dated: September 1, 2006

CLAIMS APPENDIX
PENDING CLAIMS ON APPEAL OF
U.S. PATENT APPLICATION SERIAL NO. 09/899,435

Claim 1. A telecommunication network, comprising:

a central server of an access or service provider, the central server having an interrogation part for interrogating hardware and software configurations of a plurality of terminal devices and a software transmitting part for loading at least one of software and data that is customized to the respectively detected hardware and software configuration onto one of the plurality of terminal devices;

a plurality of terminal devices, each with a predetermined hardware and software configuration, each of the plurality of terminal devices including a response transmitting part for transmitting a configuration code identifying the respective hardware and software configuration to the central server in response to an inquiry by the interrogation part, each of the plurality of terminal devices also including a software receiving part for receiving and internally storing at least one of the transferred software and data, the interrogation part and the response transmitting part being designed to interrogate the respective hardware and software configuration and to transmit the respective configuration code when at least one of the terminal device logs onto the telecommunication network, predetermined times occur, and predetermined time intervals occur; and

distributed control parts, which are distributed in both the central server and the plurality of terminal devices, the distributed control parts implementing an interactive control over the software transmitting part, and being constructed for the interactive specifying of a charging mode for at least one of downloaded software and downloaded data.

Claim 2. A telecommunications network as claimed in claim 1, further comprising: an offer memory in the central server to which the distributed control parts are connected, the offer memory being addressable via the configuration code and having a plurality of memory areas, in each of the memory areas at least one of a software and a data offer which is tuned to a separate hardware and software configuration is listed; and

wherein the distributed control parts include an offer transmitting part in the central server, the offer transmitting part for transferring contents of the respectively addressed offer memory area to the respective terminal device that has transmitted a configuration code, a transmission initiation unit in the central server, the transmission initiation unit for activating the transmitted part for loading at least one of software and data from at least one of the tuned software and the data offer, an offer display part in each of the plurality of terminal devices for displaying the memory contents of the respectively addressed offer memory area, and a requesting part in each of the plurality of terminal devices for selecting offered software and data for loading onto the terminal device, which send a request signal for at least one of desired software and data and a reject signal for unwanted software and data to the transmission initiation unit of the central server.

Claim 3. A telecommunication network as claimed in claim 2, wherein the central server further includes a reject signal storage area for terminal-device-specific storage of reject signals in association with the transmitted software and data offers, such that the reject signal storage area is allocated to the offer memory on an output side as filter so that software and data offers which are quit via a reject signal are not repeated to a same user.

Claim 4. A telecommunication network as claimed in claim 3, further comprising:
a charging mode memory in the central server to which the distributed control parts are connected, the charging mode memory being allocated to the offer memory and having at least one charging mode stored for at least one of each software offer and each data offer; and
wherein the distributed control parts include a charging mode transmitting part in the central server connected to the charging mode memory for responding to the reception of one of a configuration code and a request signal, a charging mode display part in each of the plurality of terminal devices for displaying the at least one charging mode for at least one of the offered and the selected software and the offered and the selected data, and a charging mode confirmation part in each of the plurality of terminal devices for specifying the charging mode.

Claim 5. A telecommunication network as claimed in claim 1, wherein the central server further includes a terminal device operating data memory with a plurality of memory areas for the terminal-device-specific data storage of at least one of software and data that are implemented in the plurality of terminal devices, and operating data receiving and transmitting parts connected to the terminal device operating data memory for transferring the software and data from and to the plurality of terminal devices, and wherein each of the plurality of terminal devices further includes additional operating data transmitting and receiving parts for transferring the software and data to and from the central server.

Claim 6. A telecommunication network as claimed in claim 5, wherein the operating data receiving and transmitting parts of both the central server and the plurality of terminal devices are so connected to the distributed control parts for implementing the interactive

control that the data storage in the central server occurs only upon the selection of a corresponding offer by a user of the terminal device.

Claim 7. A telecommunication network as claimed in claim 1, wherein the distributed control parts are formed as network-specific signaling parts on the basis of at least one of SIM cards, firmware, and applets/scripts.

Claim 8. A telecommunication network as claimed in claim 1, wherein the central server acts as an intermediate station in the loading of the software and the data onto a first of the plurality of terminal devices by one of a second of the plurality of terminal devices in the telecommunication network and a data terminal device in a data network which is linked to the telecommunication network.

Claim 9. A telecommunication network as claimed in claim 1, wherein the central server further includes a validation storage unit for storing at least one of validity data and authorization data in association with predetermined configuration codes as well as a comparison unit that is connected to the storage unit which compares the configuration codes that are transmitted by the plurality of terminal devices to stored configuration codes for the purpose of determining at least one of the validity of software stocks and data stocks and the usage authorization of a respective user.

Claim 10. A telecommunication network as claimed in claim 9, wherein the software stocks and the data stocks that are one of implemented in the plurality of terminal devices and

downloaded into the plurality of terminal devices include application counter elements, the central server further including an arithmetic evaluation unit for evaluating the counter statuses of the application counter elements at one of predetermined times, time intervals, and times when the relevant terminal device logs onto the telecommunication network, for the purpose of achieving a use-based charging mode.

Claim 11. A telecommunication network as claimed in claim 10, wherein the central server includes an auxiliary information transmission unit which is connected to at least one of the comparison unit and the arithmetic evaluation unit for transmitting messages to the respective terminal device relating to at least one of the validity of implemented software, the usage authorization, and the application counter status for the respective user, the plurality of terminal devices including auxiliary information reception and display units for receiving and displaying the messages.

Claim 12. A telecommunication network as claimed in claim 1, wherein the software that can be downloaded onto the plurality of terminal devices includes at least one of software components and data for implementing non-network-bound auxiliary functions of the plurality of terminal devices.

Claim 13. A telecommunication network as claimed in claim 1, wherein the software and data that can be downloaded onto the plurality of terminal devices includes software components and data for implementing auxiliary services that are available in one of the

telecommunication network and a data network that is connected to the telecommunication network.

Claim 14. A telecommunication network as claimed in claim 1, wherein the software and data that can be downloaded onto the plurality of terminal devices include update software and update data for updating software and data stocks that are stored in the plurality of terminal devices.

Claim 15. A method of operating a telecommunication network having a plurality of terminal devices of users, each with a predetermined hardware and software configuration, and a central server of an access or service provider, the method comprising the steps of:

interrogating the current hardware and software configurations of one of the plurality of terminal devices in an interrogation step at one of a time during logon onto the telecommunication network, predetermined times, and time intervals;

transmitting, in a transmission step, the current hardware and software configuration of the respective terminal device to the central server;

setting up in the central server and transmitting to the respective terminal device, based on the respectively transmitted hardware and software configuration, offer information for a user of a respective terminal device;

displaying, in the context of an interactive menu in the respective terminal device, the offer information together with one of a select and a reject request, one of a request and a reject signal of the user which is generated by the user being registered;

transmitting, together with the offer information, charging mode signals to the respective terminal device and displaying the charging mode signals in the context of the interactive menu for selection by the user, and registering a charging mode in the central server in response to a selection made by the user; and

downloading onto the respective terminal device by the central server, in response to the registered one of the request and the reject signals, at least one of software and data that are suitable for the respective terminal device and that are not already present at the respective terminal device.

Claim 16. A method of operating a telecommunication network as claimed in claim 15, the method further comprising the step of: transferring to the central server, when one of the plurality of terminal devices log onto the telecommunication network, at predetermined times, and at time intervals, software and data that is implemented in the plurality of terminal devices for the purpose of data storage, and transferring the software and data by the central server back to the plurality of terminal devices again upon the occurrence of a predetermined condition.

Claim 17. A method of operating a telecommunication network as claimed in claim 15, the method further comprising the steps of: storing at least one of the reject signals and the request signals in the central server for each individual terminal device; and generating subsequent offer information using the at least one of the stored reject signals and the stored request signals as a filter.

Claim 18. A method of operating a telecommunication network as claimed in claim 15, the method further comprising the step of: using the central server as an intermediate station in the loading of software and data onto a first of the plurality of terminal devices by one of a second of the plurality of terminal devices in the telecommunication network and a data terminal device in a data network that is linked to the telecommunication network.

Claim 19. A method of operating a telecommunication network as claimed in claim 15, the method further comprising the steps of: storing at least one of validity data and authorization data in the central server in association with predetermined configuration codes; comparing the data to configuration codes that are transmitted by the plurality of terminal devices; and outputting to the plurality of terminal devices, as a result of the comparison, data relating to at least one of the validity of software and data stocks that are stored in the plurality of terminal devices and the usage authorization of the respective user.

Claim 20. A method of operating a telecommunication network as claimed in claim 15, the method further comprising the step of: evaluating in the central server, when at least one of a terminal device logs on, predetermined times occur, and time intervals occur, counter statuses of application counter elements of the software and data stocks that are implemented in the plurality of terminal devices for the purpose of performing a use-based charging, an evaluation result being transmitted to the plurality of terminal devices.

Claim 21. A terminal device for use in a telecommunication network having a plurality of terminal devices of users, each with a predetermined hardware and software

configuration, and a central server of an accessor service provider, the terminal device comprising: a response transmitting part for transmitting a configuration code that identifies a currently implemented hardware and software configuration to the central server in response to an interrogation by the central server; distributed control parts for implementing an interactive control of the downloading of at least one of software and data from the central server; a charging mode display part for displaying at least one available charging mode for one of offered and selected software and data; and a charging mode confirmation part for specifying the charging mode.

Claim 22. A terminal device for use in a telecommunication network as claimed in claim 21, wherein the distributed control parts include an offer display part for displaying offer information that is transmitted by the central server and a requesting part for selecting at least one of software and data that is offered for downloading for the purpose of outputting at least one of a request signal and reject signal to the central server.

Claim 23. A terminal device for use in a telecommunication network as claimed in claim 21, wherein the distributed control parts include signaling parts based on at least one of SIM cards, firmware, and applets/scripts.

Claim 24. A terminal device for use in a telecommunication network as claimed in claim 21, further comprising: operating data transmitting and receiving parts for transferring at least one of software and data to and from the central server.

Claim 25. A terminal device for use in a telecommunication network as claimed in claim 21, further comprising: an auxiliary information reception and display unit for receiving and displaying messages that are transmitted on the central server side relating to at least one of the validity of software and data that are implemented in the terminal device, the usage authorization, and a use-based charge status.

EVIDENCE APPENDIX

EXHIBIT A: Office Action dated February 22, 2006

EXHIBIT B: *Kobata* (U.S. Patent No. 4,540,585), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT C: *Nakagawa* (U.S. Patent 5,835,911), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT D: *Chen et al.* (U.S. 5,797,016), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT E: *Valentine* (U.S. 6,018,654), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT F: *Pepe et al.* (U.S. Patent 5,742,668), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT G: *Shear* (U.S. Patent 4,977,594), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT H: *Shah* (U.S. Patent 6,029,065), cited by the Examiner in the Office Action dated February 22, 2006

RELATED PROCEEDINGS APPENDIX

None

EXHIBIT A



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Add: as COMMISSIONER FOR PATENTS
P.O. Box 1459
Alexandria, Virginia 22313-1459
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/899,435	07/05/2001	Gero Offer	112740-257	3369
29177	7590	02/22/2006	EXAMINER	
BELL, BOYD & LLOYD, LLC			CHEA, PHILIP J	
P. O. BOX 1135			ART UNIT	
CHICAGO, IL 60690-1135			PAPER NUMBER	
			2153	

DATE MAILED: 02/22/2006

Due: 5-22-06

Please find below and/or attached an Office communication concerning this application or proceeding.

RECEIVED
BELL, BOYD & LLOYD
INTELLECTUAL PROPERTY DOCKET

FEB 27 2006

ATTY

BOOKET #

PAK-P22
112740-0257

Office Action Summary

Application No.

09/899,435

Applicant(s)

OFFER, GERO

Examiner

Philip J. Chea

Art Unit

2153

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 25 November 2005.
- 2a) ☒ This action is FINAL. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-25 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-25 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 05 July 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☒ All b) ☐ Some * c) ☐ None of:
1. ☒ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☐ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____.
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date _____.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: _____.

DETAILED ACTION

This Office Action is in response to an Amendment filed November 25, 2005. Claims 1-25 are currently pending. Any rejection not set forth below has been overcome by the current Amendment.

Information Disclosure Statement

1. The Examiner acknowledges the corrected reference number.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1-4,14,21,22,24 rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata (US 6,058,418), and further in view of Nakagawa et al. (US 5,838,911), herein referred to as Nakagawa.

As per claims 1, 21, Kobata discloses a telecommunication network, as claimed, comprising:

central server of an access or service provider, the central server having an

interrogation part for interrogating hardware and software configurations of a plurality of terminal devices and a software transmitting part for loading at least one of software and data that is customized to the respectively detected hardware and software configuration onto one of the plurality of terminal devices (see Fig. 1, Infrastructure Data and columns 4 and 5, lines 57-67 and 1-18);

a plurality of terminal devices, each with a predetermined hardware and software configuration, each of the plurality of terminal devices including a response transmitting part for transmitting a configuration code identifying the respective hardware and software configuration to the central server in response to an inquiry by the interrogation part (see Fig. 1, Infrastructure Data with Serial No. and column 4, lines 57-67), each of the plurality of terminal devices also including a software receiving part for receiving and internally storing at least one of the transferred software and data (see column 5, lines 1-

Art Unit: 2153

18), the interrogation part and the response transmitting part being designed to interrogate the respective hardware and software configuration and to transmit the respective configuration code when at least one of the terminal device logs onto the telecommunication network, predetermined times occur, and predetermined time intervals occur (see column 4, lines 57-67); and

distributed control parts, which are distributed in both the central server and the plurality of terminal devices, the distributed control parts implementing an interactive control over the software transmitting part.

Although the system disclosed by Kobata shows substantial features of the claimed invention (discussed above), it fails to disclose distributed control parts implementing an interactive control over the software transmitting part, and being constructed for the interactive specifying of a charging mode for at least one of downloaded software and downloaded data.

Nonetheless, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata, as evidenced by Nakagawa et al.

In an analogous art, Nakagawa discloses a software distributing system for updating a client with current software (see Abstract) and further showing an interactive control over transmitting software (see column 67, lines 50-60), and interactively specifying a charging mode for at least one of downloaded software and downloaded data (see column 68, lines 17-29).

Given the teaching of Nakagawa, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata by employing interactive control and a charging mode, such as disclosed by Nakagawa, in order to allow the user to choose certain programs they want and different methods of payment that may suit their needs.

As per claim 2, Kobata in view of Nakagawa further disclose an offer memory in the central server to which the distributed control parts are connected, the offer memory being addressable via the configuration code and having a plurality of memory areas, in each of the memory areas at least one of a software and a data offer which is tuned to a separate hardware and software configuration is listed (see Kobata Fig. 5); and

wherein the distributed control parts include an offer transmitting part in the central server, the offer transmitting part for transferring contents of the respectively addressed offer memory area to the respective terminal device that has transmitted a configuration code, a transmission initiation unit in the central server, the transmission initiation unit for activating the transmitting part for loading at least one of software and data from at least one of the tuned software and the data offer (see Kobata Fig. 5 [112]), an offer display part in each of the plurality of terminal devices for displaying the memory contents of the respectively addressed offer memory area, and a requesting part in each of the plurality of terminal devices for selecting offered software and data for loading onto the terminal device (see Kobata column 5, lines 41-44), which send a request signal for at least one of desired software and data and a reject signal for unwanted software and data to the transmission initiation unit of the central server (see Kobata column 5, lines 41-44, where accepting certain version implies an accept signal for the version wanted and a reject signal for the version unwanted).

As per claim 3, although Kobata in view of Nakagawa discloses substantial features of the claimed invention (discussed above), it fails to directly disclose a filtering method based on previously rejected software by the terminal device. However, it would have been obvious to a person having ordinary skill in the art to stop offering software, which the terminal device has rejected. The desirability and advantages of having such a filtering means would be for the convenience of the user since the user would not have to continually decline the same software they do not want, analogous to a software system for allowing a user to stop checking for software updates by checking a box.

As per claim 4, Kobata in view of Nakagawa further disclose a charging mode memory in the central server to which the distributed control parts are connected, the charging mode memory being allocated to the offer memory and having at least one charging mode stored for at least one of each offered software item (see Nakagawa column 27, lines 19-26); and

wherein the distributed control parts include a charging mode transmitting part in the central server connected to the charging mode memory for responding to the reception of the one of a configuration code and a request signal, a charging mode display part in each of the plurality of terminal devices for displaying the at least one charging mode for at least one of the offered and the selected

Art Unit: 2153

software and the offered and the selected data, and a charging mode confirmation part in each of the plurality of terminal devices for specifying the charging mode (see column 68, lines 17-29).

As per claim 14, Kobata in view of Nakagawa further disclose that the software and data that can be downloaded onto a plurality of terminal devices include update software and update data for updating software and data stocks that are stored in the plurality of terminal devices (see Nakagawa column 67, lines 23-29).

As per claim 22, Kobata in view of Nakagawa further disclose an offer display part for displaying offer information that is transmitted by the central server and a requesting part for selecting at least one of software and data that is offered for downloading for the purpose of outputting at least one of a request signal and reject signal to the central server (see Kobata column 5, lines 41-44, where accepting certain version implies an accept signal for the version wanted and a reject signal for the version unwanted).

As per claim 24, Kobata in view of Nakagawa further disclose operating data transmitting and receiving parts for transferring at least one of software and data to and from the central server (see Kobata columns 4 and 5, lines 57-67 and 1-18).

4. Claims 5,6 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata in view of Nakagawa as applied to claim 1 above, and further in view of Chen et al. (U.S. 5,797,016), herein referred to as Chen.

As per claim 5, although Kobata in view of Nakagawa discloses substantial features of the claimed invention (discussed above), he fails to directly disclose a system where software can be stored and transferred to and from the server and terminal device. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Nakagawa, as evidenced by Chen.

In an analogous art, Chen discloses a system with a backup server for storing files for current software configurations (column 3, lines 32-34) from a terminal device (column 4, lines 39-43, where files can also be transmitted from the server to the terminal device [column 2, lines 63-67]), as claimed above.

Art Unit: 2153

Given the teaching of Chen, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Nakagawa by employing a server that can store files from terminal devices, such as disclosed by Chen, in order to backup current software configurations incase new installations are not successful, the old software can be restored (column 3, lines 32-34).

As per claim 6, Kobata in view of Nakagawa in view of Chen further disclose that the operating data receiving and transmitting parts of both the central server and the plurality of terminal devices are so connected to the distributed control parts for implementing the interactive control that the data storage in the central server occurs only upon the selection of a corresponding offer by a user of the terminal device (column 4, lines 37-43, creation of the back job is considered conditional on the part of the administrator).

5. Claims 7,12,13,23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata in view of Nakagawa as applied to claims 1 and 21 above, and further in view of Valentine (U.S. 6,018,654).

As per claims 7 and 23 although Kobata in view of Nakagawa discloses substantial features of the claimed invention (discussed above), he fails to directly disclose network-specific signaling parts on the basis of a SIM card. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Nakagawa, as evidenced by Valentine.

In an analogous art, Valentine disclose a system with a server with a transmitting part for transmitting software to mobile devices [Figure 3, (170, 20)] using signaling parts on the basis of a SIM card (column 3, lines 7-12).

Given the teaching of Valentine, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Nakagawa by employing signaling parts on the basis of a SIM card, such as disclosed by Valentine, in order to interactively download software (see Fig. 4, and columns 4, lines 15-27).

As per claim 12, although Kobata in view of Nakagawa discloses substantial features of the claimed invention (discussed above), he fails to directly disclose software for implementing non-network-

bound auxiliary functions. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Nakagawa, as evidenced by Valentine

Valentine discloses a system with a server with a transmitting part for transmitting software to mobile devices [Figure 3, (170, 20)] where the software being transmitted is non-network bound (column 1, lines 44-52, where tone data is considered non-network bound). Given the teaching of Valentine, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Nakagawa by employing software for implementing non-network-bound functions such as ring tones, such as disclosed by Valentine, in order to provide users with entertainment or, in the case of ring tones, help them determine the caller without picking up the phone (column 1, lines 29-38).

As per claim 13, it would have been obvious to a person having ordinary skill in the art to offer software for implementing auxiliary services, since the type of software is simply a choice left up to the designer. The desirability and advantages of implementing auxiliary services would be for conveniently allowing the user to connect to the Internet using well-known applications such as a web browser or IM chat client.

6. Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata in view of Nakagawa as applied to claims 1 and 15 above, and further in view of Pepe et al. (U.S. 5,742,668).

Although Kobata in view of Nakagawa disclose substantial features of the claimed invention (discussed above), he fails to directly disclose a central server, which acts as an intermediate station between two terminal devices for loading data from one terminal device to another. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Nakagawa, as evidenced by Pepe et al.

In an analogous art, Pepe et al. disclose a telecommunications network with a central server used for transmitting data between devices (column 13, lines 29-39). Devices include cellular phones, PDAs, and email from workstations [Figure 1, (32, 30, 22)].

Given the teaching of Pepe et al., a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Nakagawa by employing a central server to act as an intermediate station to load software between devices, such as disclosed by Pepe et al., in order to communicate messages between devices from anywhere at anytime (column 1, lines 42-43).

7. Claims 9-11,25 rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata in view of Nakagawa as applied to claims 1,21 above, and further in view of Shear (U.S. 4,977,594).

As per claim 9, although Kobata in view of Nakagawa discloses substantial features of the claimed invention (discussed above), he fails to directly disclose determining the validity of software and usage authorization. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Nakagawa, as evidenced by Shear.

In an analogous art, Shear discloses a system for distributing data with a validation storage unit for storing authorization data, and determining the validity of data of terminal devices (column 14; lines 29-36).

Given the teaching of Shear, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Nakagawa by employing the validation storage unit, such as disclosed by Shear, in order to bill the user according to the amount of resources used (column 3, lines 52-59). In a software/data distributing environment where resources are not free, it would be advantageous for the supplier of the software/data to keep track of the software being deployed in order to charge the appropriate amount to the user.

As per claim 10, Kobata in view of Nakagawa in view of Shear further disclose that the software stocks and data stocks that are one of implemented in the plurality of terminal devices and downloaded into the plurality of terminal devices include application counter elements, the central server further including an arithmetic evaluation unit for evaluating the counter statuses of the application counter elements at one of predetermined times, time intervals, and times when the relevant terminal device logs onto the telecommunication network, for the purpose of achieving a use-based charging mode (see Shear column 13, lines 47-60).

As per claims 11,25, Kobata in view of Nakagawa in view of Shear further disclose that the server includes an auxiliary information transmission unit which is connected to at least one of the comparison unit and the arithmetic evaluation unit for transmitting messages to the respective terminal device relating to at least one of the validity of implemented software, the usage authorization, and the application counter status for the respective user, the plurality of terminal device including auxiliary information reception and display units for receiving and displaying the messages (see Shear column 15, lines 33-47).

8. Claims 15,17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata, and further in view of Shah (US 6,029,065) and further in view of Nakagawa (US 5,838,911).

As per claim 15, Kobata discloses a method of operating a telecommunication network having a plurality of terminal devices of users, each with a predetermined hardware and software configuration, and a central server of an access or service provider, the method comprising the steps of:

interrogating the current hardware and software configurations of one of the plurality of terminal devices in an interrogation step at one of a time during logon onto the telecommunication network, predetermined times, and time intervals;

transmitting, in a transmission step, the current hardware and software configuration of the respective terminal device to the central server;

setting up in the central server and transmitting to the respective terminal device, based on the respectively transmitted hardware and software configuration, offer information for a user of a respective terminal device; and

downloading onto the respective terminal device by the central server, in response to the registered one of the request and the reject signals, software and/or data that are suitable for the respective terminal device and that are not already present at the terminal device (see column 5, lines 1-18).

Although the system disclosed by Kobata shows substantial features of the claimed invention (discussed above), it fails to disclose displaying, in the context of an interactive menu in the respective

terminal device, the offer information together with one of a select and a reject request, one of a request and a reject signal of the user which is generated by the user being registered.

Nonetheless, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata, as evidenced by Shah.

In an analogous art, Shah discloses a system where a mobile station can select a desired feature available to the mobile station (see Abstract) further, further showing an interactive menu in the respective terminal device with one of a select and reject request, one of a request and a reject signal of the user which is generated by the user being registered (see column 9, lines 54-61).

Given the teaching of Shah, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata by employing an interactive menu, such as disclosed by Shah, in order to allow a user to easily pick which features they want to implement.

Although the system disclosed by Kobata in view of Shah shows substantial features of the claimed invention (discussed above), it fails to disclose displaying the charging mode signals in the context of the interactive menu for selection by the user, and registering a charging mode in the central server in response to a selection made by the user.

Nonetheless, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Shah, as evidenced by Nakagawa et al.

In an analogous art, Nakagawa discloses a software distributing system for updating a client with current software (see Abstract) and further specifying a charging mode for at least one of downloaded software and downloaded data (see column 68, lines 17-29).

Given the teaching of Nakagawa, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Shah by employing interactive control and a charging mode, such as disclosed by Nakagawa, in order to allow the user to choose different methods of payment that may suit their needs.

As per claim 17, although Kobata in view of Shah in view of Nakagawa discloses substantial features of the claimed invention (discussed above), it fails to directly disclose a filtering method based on previously rejected software by the terminal device. However, it would have been obvious to a person

having ordinary skill in the art to stop offering software, which the terminal device has rejected. The desirability and advantages of having such a filtering means would be for the convenience of the user since the user would not have to continually decline the same software they do not want, analogous to a software system for allowing a user to stop checking for software updates by checking a box.

9. Claims 19,20 rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata in view of Shah in view of Nakagawa as applied to claims 1,21 above, and further in view of Shear (U.S. 4,977,594).

As per claim 19, although Kobata in view of Shah in view of Nakagawa discloses substantial features of the claimed invention (discussed above), he fails to directly disclose determining the validity of software and usage authorization. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Shah in view of Nakagawa, as evidenced by Shear.

In an analogous art, Shear discloses a system for distributing data with a validation storage unit for storing authorization data, and determining the validity of data of terminal devices (column 14, lines 29-36).

Given the teaching of Shear, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Shah in view of Nakagawa by employing the validation storage unit, such as disclosed by Shear, in order to bill the user according to the amount of resources used (column 3, lines 52-59). In a software/data distributing environment where resources are not free, it would be advantageous for the supplier of the software/data to keep track of the software being deployed in order to charge the appropriate amount to the user.

As per claim 20, Kobata in view of Shah in view of Nakagawa in view of Shear further disclose evaluating in the central server, when at least one of a terminal device logs on, predetermined times occur, and time intervals occur, counter statuses of application counter elements of the software and data stocks that are implemented in the plurality of terminal devices for the purpose of performing a use-based charging, an evaluation result being transmitted to the plurality of terminal devices (see Shear column 13, lines 47-50).

10. Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata in view of Shah in view of Nakagawa as applied to claim 15 above, and further in view of Chen et al. (U.S. 5,797,016), herein referred to as Chen.

Although Kobata in view of Shah in view of Nakagawa discloses substantial features of the claimed invention (discussed above), he fails to directly disclose a system where software can be stored and transferred to and from the server and terminal device. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Shah in view of Nakagawa, as evidenced by Chen.

In an analogous art, Chen discloses a system with a backup server for storing files for current software configurations (column 3, lines 32-34) from a terminal device (column 4, lines 39-43, where files can also be transmitted from the server to the terminal device [column 2, lines 63-67]), as claimed above. Given the teaching of Chen, a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Shah in view of Nakagawa by employing a server that can store files from terminal devices, such as disclosed by Chen, in order to backup current software configurations incase new installations are not successful, the old software can be restored (column 3, lines 32-34).

Claim 18 rejected under 35 U.S.C. 103(a) as being unpatentable over Kobata in view of Shah in view of Nakagawa as applied to claims 1 and 15 above, and further in view of Pepe et al. (U.S. 5,742,668). Although Kobata in view of Shah in view of Nakagawa disclose substantial features of the claimed invention (discussed above), he fails to directly disclose a central server, which acts as an intermediate station between two terminal devices for loading data from one terminal device to another. However, these features are well known in the art and would have been an obvious modification of the system disclosed by Kobata in view of Shah in view of Nakagawa, as evidenced by Pepe et al.

In an analogous art, Pepe et al. disclose a telecommunications network with a central server used for transmitting data between devices (column 13, lines 29-39). Devices include cellular phones, PDAs, and email from workstations [Figure 1, (32, 30, 22)].

Given the teaching of Pepe et al., a person having ordinary skill in the art would have readily recognized the desirability and advantages of modifying Kobata in view of Shah in view of Nakagawa by employing a central server to act as an intermediate station to load software between devices, such as disclosed by Pepe et al., in order to communicate messages between devices from anywhere at anytime (column 1, lines 42-43).

Response to Arguments

11. Applicant's arguments filed November 25, 2005 have been fully considered but they are not persuasive.

(A) Applicant contends that in the claimed invention the interrogation of the hardware and software configurations of the terminal devices is initiated by the server.

(B) Applicant contends Nakagawa contains no distributed control means that are arranged in both the server and the client.

(C) Applicant contends there is no teaching, suggestion or motivation for one of ordinary skill in the art to combine Kobata/Nakagawa and Kobata/Shah references.

In considering (A), the Examiner believes that the claim is unclear as to whether the server initiates the communication. It is understood that the client responds to an inquiry by the interrogation part, but the claim does not specifically state if the server or client is the first to engage. The Examiner invites the Applicant to clearly indicate that the server is the first to initiate communication. Kobata is still pertinent even if initial communication is originated from the server because as shown in column 4, lines 57-column 5, line 18, a server sends out a package to client to check the infrastructure data of the client

(i.e. CPU info HD space, etc.), then pushes content onto the client which is suitable based on the infrastructure.

In considering (B), the Examiner respectfully disagrees. The system of Nakagawa requires interaction of both the server and the client. Without both, a client would not be able to request for purchase information at all. Examiner invites the applicant to elaborate on how the client executes the interactive specifying.

In considering (C), the Examiner respectfully disagrees. In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, both systems require the knowledge of a clients software/hardware configuration in order to deliver content accordingly. A person of ordinary skill in the art would recognize the similarity between the systems regarding the functional pieces of hardware.

Conclusion

12. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Art Unit: 2153


Any inquiry concerning this communication or earlier communications from the examiner should be directed to Philip J. Chea whose telephone number is 571-272-3951. The examiner can normally be reached on M-F 7:00-4:30 (1st Friday Off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Glenn Burgess can be reached on 571-272-3949. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Philip J Chea
Examiner
Art Unit 2153

PJC 2/9/06



GLENN B. BURGESS
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

EXHIBIT B

United States Patent [19]

Priegnitz

[11] Patent Number: 4,540,585

[45] Date of Patent: Sep. 10, 1985

- [54] **FOOD PRODUCTS CONTAINING
α-AMYLASE AND PROCESS**
- [75] Inventor: Ronald D. Priegnitz, Algonquin, Ill.
- [73] Assignee: The Quaker Oats Company, Chicago, Ill.
- [21] Appl. No.: 517,878
- [22] Filed: Jul. 27, 1983

3,745,021	7/1973	Van Middlesworth et al. ...	426/646
3,950,543	4/1976	Buffa et al.	426/18
3,968,255	7/1976	Haas et al.	426/637
4,190,679	2/1980	Coffee et al.	426/805
4,299,848	11/1981	De Stefanis et al.	426/64
4,320,151	3/1982	Cole	426/18
4,344,969	8/1982	Youngquist et al.	426/18
4,376,129	3/1983	Plukovich et al.	426/635
4,391,829	7/1983	Spradlin et al.	426/28
4,393,085	7/1983	Spradlin et al.	426/28
4,418,086	11/1983	Marino et al.	426/805

Related U.S. Application Data

- [63] Continuation of Ser. No. 262,252, May 11, 1981, abandoned.

- [51] Int. Cl.¹ A21D 13/06; A23K 1/00;
A23L 3/34
- [52] U.S. Cl. 426/28; 426/64;
426/321; 426/549; 426/516; 426/805
- [58] Field of Search 426/18, 28, 61, 641,
426/321, 805, 64, 2, 496, 549, 516

References Cited

U.S. PATENT DOCUMENTS

2,615,810	10/1952	Stone	426/64
3,202,514	8/1965	Burgess et al.	426/532
3,617,300	11/1971	Borochoff et al.	426/28

FOREIGN PATENT DOCUMENTS

1544499 4/1976 United Kingdom .

Primary Examiner—Raymond Jones
Assistant Examiner—Elizabeth C. Weimar
Attorney, Agent, or Firm—Karen E. Ayd; Daniel W. Latham

ABSTRACT

[57] A pet food product containing at least one amylaceous ingredient maintains or improves its soft texture during storage when the enzyme α-amylase is added thereto and processed using heat.

5 Claims, No Drawings

FOOD PRODUCTS CONTAINING α -AMYLASE AND PROCESS

This application is a continuation of application Ser. No. 262,252, filed May 11, 1981, now abandoned.

BACKGROUND OF THE INVENTION

This invention relates to food products and more particularly to food products of an amylaceous character containing heat stable α -amylase which is added in amounts sufficient to maintain or improve the soft texture after a heat treatment.

While it is well recognized that many features contribute to the taste of a food, it is equally well recognized that an important taste feature of a food is its texture, or, the relative softness of the food. A food that feels hard or conversely, extremely soft or mushy may not be acceptable, the desired texture lying generally somewhere in the intermediate range. Foods such as meat or meat substitutes are usually expected to be relatively soft. Some products though initially of a soft texture, such as bread, are known to stale and become hard upon storage. No imagination is needed to realize that with a wide variety of foods, texture or, retaining a soft texture, can be a large problem.

Even though it is possible to manufacture a food product having initially the desired soft texture, staling and the like are factors encountered on storage which can have an adverse effect on texture.

Because such hardness and change in texture is inherently undesirable in most food products, it thus becomes clearly desirable to prevent this change in texture or reduce the texture deterioration in a stored food.

Procedures are known in the prior art for achieving a food having a soft texture by addition of such ingredients as humectants. In some cases, a food which is manufactured with a soft texture has such a low amount of texture deterioration that storage is highly feasible for that food without a substantial loss in texture. Even in these cases, there is some loss of texture which can become detrimental if the storage is overly extended. The components generally used to achieve the desired texture are costly and can have a somewhat adverse effect on palatability.

Besides the use of humectants as taught in U.S. Pat. No. 3,202,514 to Burgess, another way to achieve and maintain a soft texture is through the use of certain emulsifiers which tend to complex with the starch, thereby retarding starch degradation. However emulsifiers only slow down the rate of staling, and do not completely inhibit it.

It has been known in the past to add an amylase enzyme to bakery products such as breads, to achieve a softening of the bread texture. However, in the baking process, the bread dough is cooked at a sufficiently high temperature for a sufficiently long time to assure inactivation of the enzyme during baking. Thus no texturization of the bread can occur after baking, and the bread also will become hard or stale with time.

SUMMARY OF THE INVENTION

Therefore, an object of the subject invention is a food product and a process for preparing a soft food product having a texture which can be maintained during storage.

A still further object of the subject invention is a food product, such as a pet food, having improved texture

upon storage while avoiding adverse effects on its palatability.

These and other objects of this invention are accomplished by adding an effective amount of α -amylase containing material to a food product, processing the food product using heat without completely inactivating the α -amylase and then holding such product in storage such that the soft texture is maintained or improved upon storage.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

A food product containing an amylaceous ingredient, such as cereal grains, maintained or improved its texture due to the addition of an effective amount of active α -amylase.

The amount of α -amylase suitable for use in this invention varies substantially depending on the type of food product in which it is to be used, and can be easily determined by a person having ordinary skill in the art. Generally speaking, from a trace to about 5000 SKB units of α -amylase is added for each kilogram of starch in the food being made and is present in the food after processing. More preferably, from about 0.25 SKB units to about 2000 SKB units per kilogram of starch in the product is in the food after processing. Even more preferably, about 0.5 SKB units per kilogram of product to about 1000 SKB units per kilogram of starch in the product may be added to achieve the appropriate levels in the food product after processing. The reference to SKB units is well known in the art as a definition for amylase activity as set forth in Sanstedt, et al., Cereal Chemistry, Vol. 16, page 712, (1939). A factor to be considered in determining the effective amount of enzyme to be used to achieve a desired texture in a food product is the moisture content of the food product. Unless the food product contains over fifteen percent moisture, only minimal or no texturizing effect will be noted. Based on current observations, it would appear that appreciable enzymatic activity of the α -amylase occurs only at moisture levels above about fifteen percent.

α -Amylase is a well known enzyme, having an IUB number of 3.2.1.1, which defines α -amylase in an internationally accepted manner. It is also known as 1,4- α -D glucan glucanohydrolase. α -Amylase is generally present in all amylaceous materials at low levels and, by previous practices, was generally reduced to a low ineffective concentration or activity level by baking. According to the subject invention, α -amylase is now added to the food product in quantities, that will assure a sufficiently high activity level after processing to maintain or improve the texture of the food product. In this manner texture deterioration in the food product is avoided even after storage.

While not completely understood, it is theorized that within an amylaceous product the active α -amylase present reacts to hydrolyze the starch, thus reducing the amount of long starch chains available for retrogradation, and resulting in a soft texture throughout the entire product. However the starch does not become susceptible to the action of the enzyme until gelatinization of the starch occurs at temperatures destructive to the α -amylase. As a result, the heat processing of a food product generally inactivates the enzyme, allowing a staling process to go forward unless other measures are taken. For instance, the amylase content of a bread dough may affect the texture of the bread initially, however the

softness of the bread begins to deteriorate as soon as the baking is completed.

The rate and degree of softening of the food is dependent upon the activity of the enzyme used, the presence and amount of free moisture, enzyme, chloride ion, calcium, the pH and temperature of the product. The α -amylase may be derived from bacterial, fungal, animal, cereal or vegetable sources. Bacterial amylase is preferred for the purpose of this invention because it is readily available, can be readily adapted for use in foods, and is more heat stable. α -Amylase generally is active in a pH range of about 2 to 11 with a moisture content of over 10-15%. The preferred pH range for α -amylase activity is in the 4 to 8 range. Most foods contain sufficient amount of chloride, in the form of salt and calcium to activate the enzyme. If calcium is not present in the food, any suitable source from edible salts may be added, such as calcium phosphate and calcium sulfate.

Effective processing of an amylaceous food product generally depends on temperature and time, and must be sufficient to cook the food and gelatinize the starch, yet insufficient to inactivate the α -amylase. In other words, for an extruded product the temperature and residence time in an extruder must be sufficient to cook the food and gelatinize the starch, yet insufficient to inactivate the α -amylase. Processing temperatures of food containing α -amylase can range up to 150° C. for short periods of time with the effective amounts of the added α -amylase remaining active. However, it is more preferable to operate at process temperatures between 75° C. and 110° C., which is an effective temperature range for achieving most food processing and stabilization. It is critical that the α -amylase not be totally or substantially inactivated by such conditions as overly high temperatures or long retention times during processing.

Generally speaking, the higher the temperature of the extruder, the lower the residence time the food is permitted to be in the extruder. Preferably, from entry into the extruder to exit from the extruder, the time frame may be from 5 seconds to 15 minutes, dependent chiefly on the extruder used and the output rate.

Comparison tests of food with and without α -amylase added show that extruded foods with α -amylase added are at least 5 percent softer and sometimes 10 to 50 percent softer than the extruded foods without α -amylase added, when stored for the same period of time under the same conditions. Thus food products without the α -amylase added, that is, containing only natural levels of the enzyme, experienced a staling or hardening effect, while the food products with the additional α -amylase were at least as soft as the original texture, and sometimes softer as measured by the standard methods used for determining the softness of the food, i.e., the Kramer Shear Press method or alternatively by the Instron method, both known in the art.

While this concept is applicable to any type of food, it is especially applicable to semi-moist foods because a soft and moist texture is of considerable importance in semi-moist human or pet foods. While in the past the desired texture has been accomplished by adding considerable amounts of plasticizers or humectants, these ingredients are now not necessary to provide a soft texture. The presence of the added α -amylase with the other ingredients during processing provides the desired soft texture without added humectants or plasticizers. As stated above, the enzyme is believed to hy-

drolyze the starch thus reducing the long starch chain available for retrogradation. The resultant product is less apt to stale because most of the starch components not already in the crystalline state are hydrolyzed and cannot retrograde. The hydrolysis of the starch does not take place appreciably until the starch is gelatinized. Of course, if pregelatinized starch sources are used rather than unprocessed starch sources, enzymatic hydrolysis of the starch will take place in the product dough. Thus, by adding the α -amylase and maintaining its activity through a judicious choice of processing parameters with regard to temperature and time as set forth above, the α -amylase can hydrolyze the gelatinized starch in a controlled manner, even during storage. In this fashion, the product cannot only retain its initial soft texture, but can even become softer with the passage of time.

Another advantage in adding α -amylase is an improvement in palatability upon storage. It is theorized that the enzymatic hydrolysis increases the sweetness of the product during storage, since dextrins are formed, in addition to its texture softening action. It is also well known that both humans and pets prefer a soft, moist, and sweet product. The α -amylase produces just such effects on storage.

If, indeed, a pet food is desired to be formed using α -amylase, it must contain a sufficient amount of protein from either an animal source, a cereal source, a vegetable source, or mixtures thereof to provide the protein requirements as set forth by the National Research Council. Also, other ingredients to provide a suitable pet food may be included, as known in the art.

As indicated, the protein source for a pet food product is either a vegetable protein source, a cereal protein source, an animal derived protein source, or a combination thereof. The critical point of the protein source is that it must provide the nutritional and legal requirements for the protein in the product. Generally speaking, the protein content in the pet food must be at least about 5 percent to about 50, and more preferably 10 percent to 40 percent on a dry basis. Protein levels are critical depending on the type of pet food being fed. A dog food protein content is advantageously above 22 percent by weight of the pet food on a dry basis while a cat food protein content is advantageously above 28 percent by weight on a dry basis which are recommendations made by National Research Council for dogs and cats respectively to have a completely nutritional food.

The starch content of the food product of the subject invention may vary from 10% to 60% of the total ingredients. A starch content less than 10% may not provide a significant enough texturizing effect on the food product to be noticeable, while more than 60% starch can result in an excessively soft texture. Such starches will usually originate predominantly from cereal grains; however other sources, such as tubers and legumes, may be used.

The subject invention has been found particularly useful in maintaining and enhancing the texture of semi-moist pet foods. A typical semi-moist pet food of about 15 percent to about 40 percent moisture content has about 3 percent to about 65 percent of a protein source, which may comprise meat, vegetable, meat meals or mixtures thereof. A preservative system comprising sugars, edible acids and antimicrobial agents may be utilized as described in U.S. Pat. No. 3,202,514. Other

additives, such as vitamins, minerals, colorants, flavorants, and salts may be added as desired.

While the invention is now clearly disclosed as to how to make and use the invention, the following Examples are presented to guarantee an understanding of the invention without necessarily limiting the invention. All parts and percentages are by weight unless otherwise specified.

EXAMPLE I

EXPANDED SEMI-MOIST PET FOOD		
Ingredients	(A) (Without) (α -Amylase)	(B) (With) (α -Amylase)
	18.0 pts	18.0 pts
Meat and Bone Meal	18.0 pts	18.0 pts
Wheat Flour	9.0 pts	9.0 pts
Oatmeal	9.0 pts	9.0 pts
Soy Flour	5.0 pts	5.0 pts
Propylene Glycol	5.0 pts	5.0 pts
Ground Yellow Corn	4.5 pts	4.5 pts
Animal Fat	3.0 pts	3.0 pts
Oat Flour	1.0 pts	1.0 pts
Phosphoric Acid	0.1 pts	0.1 pts
Potassium Sorbate	1.4 pts	1.4 pts
Vitamins, Minerals, Color	—	30 SKB Units/ kg. starch
α -Amylase*	—	26.0 pts
Water	26.0 pts	100.0 pts
	100.0 pts	100.0 pts

*Available from Miles Laboratories as HT-1000

The above products were processed on a Bonnot extruder at 225° F. The resulting products each had bulk densities of 31 lbs./ft.³ and moisture contents of 30%. Both samples were identical in appearance and texture when first made. However, after 8 months of storage at both 73° F. and 100° F., the product containing the α -amylase enzyme was considerably softer. Both stored products were measured for softness on the Kramer Shear Press with the following results:

Product	SAMPLE Age	Maximum Peak Height at 73° F. Storage	Maximum Peak Height at 100° F. Storage
		89.7	94.6
A (no enzyme)	8 months	89.7	94.6
B (with enzyme)	8 months	58.2	51.5

The data from the Kramer Shear Press readings confirmed the observed softer texture of the product containing α -amylase. Also, the higher storage temperature seemed to increase the amount of softening by the enzyme, which was apparent both by observation and by the Kramer Shear Press readings.

EXAMPLE II

EXPANDED SEMI-MOIST PET FOOD

A formulation similar to that in Example I was used to test the texture softening action of Miles Taka-Therm® α -amylase, an amylase having a different activity level than Miles HT-1000. As in Example I, the only difference was that one contained 100 SKB units/kg-starch Miles Taka-Therm® α -amylase (D) and the other (C) did not.

Both products were processed at 230° F. in an extruder, and had finished product moisture contents of 30%, being expanded to a density of 32 lbs./ft.³. No differences in product texture were observed initially, but after 3½ months of ambient storage, the product containing the α -amylase was considerably softer. The

following measurements were made on the stored products using a shear cell on an Instron Universal Testing Machine, Model II:

Shear Cell Conditions		Sample
Sample Size: 40 gm		C, no enzyme
Stroke Length: 6 gm		D, α -amylase
Stroke Speed: 10 cm./min.		
Full Scale: 50 Kg. full scale		
Product Measurements		Average Peak Height
Age		
C, 3½ months		410.5 cm
D, 3½ months		263.5 cm

As can be seen, the Instron measurements confirmed the improved softness observed in the stored sample containing α -amylase enzyme.

EXAMPLE III

NON-EXPANDED SEMI-MOIST PET FOOD		
Ingredients	E Without α -Amylase	F With α -Amylase
	25.0 pts	25.0 pts
Meat By-Products	25.0 pts	25.0 pts
Wheat Flour	19.0 pts	19.0 pts
Soybean Flour	6.0 pts	6.0 pts
Corn Syrup	5.0 pts	5.0 pts
Propylene Glycol	4.0 pts	4.0 pts
Corn Flour	1.0 pts	1.0 pts
Phosphoric Acid	0.1 pts	0.1 pts
Potassium Sorbate	5.9 pts	5.9 pts
Vitamins, Minerals, Color, etc.	—	600 SKB Units/ kg. Starch
α -Amylase*	—	9.0 pts
Water	9.0 pts	100.0 pts
	100.0 pts	100.0 pts

*Available from Miles Laboratories as HT-1000

The above products were processed on a Bonnot extruder at 190° F. The products were made in a strand form at a finished product moisture content of 30%, and were tested for palatability initially and after 9 months of ambient storage. After nine months the observed softness of the product F (with α -amylase) was greater than product E (without α -amylase). A standard 2 bowl preference method was used to obtain the following results on palatability:

Product	% Consumption	Level of Statistical Significance
	Initial 2-Pan Test	
E, (without α -amylase)	46	(Not Significant)
F, (with α -amylase)	54	
	Nine Month Storage 2-Pan Test	
E, (without α -amylase)	36	99% (Highly Significant)
F, (with α -amylase)	64	

According to the above data there was no initial significant difference in the palatability of the products with or without α -amylase. However, after nine months of storage, not only was the product with enzyme noticeably softer, but it was significantly more palatable than the same product without the enzyme.

EXAMPLE IV

The procedure of Example III is applied to the following formulation:

Ingredients	Formula
Granola Mix (Oat flakes, Wheat flakes, Nuts, Honey, etc.)	45 pts
Sucrose	10 pts
Glycerol	10 pts
Shortening	5 pts
Vitamins, Minerals, Flavor, Phosphoric Acid, and Potassium Sorbate	2 pts
α -Amylase	20. SKB Units/kg. Starch
Water	<u>28 pts</u>
	100.

The product of Example 4 results in a granola snack bar which maintains a soft product texture over an extended shelflife without becoming hard or crumbly in texture. The α -amylase maintains the desired texture, but does not hydrolyze to the point where the product becomes too soft.

The above examples show that α -amylase is useful as a texturizing agent, not only preventing the degradation of a soft texture of a food product but even contributing to the improvement of the texture of the food product on storage. As stated and as demonstrated in the examples, the subject invention applies to all food products having amylaceous ingredients, regardless of whether the food product is for humans or for pets.

While the invention has been described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted for elements thereof without departing from the scope of the invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the invention without departing from the essential scope thereof. Therefore, it is intended that the invention not be limited to the particular embodiment disclosed as the best mode contemplated for carrying out this invention, but that the invention will include all

embodiments falling within the scope of the appended claims.

I claim:

1. In a soft, semi-moist pet food product consisting of soft, cohesive, extrusion cooked particles having 10-60% starch, 22-30% protein, 5-35% water soluble solutes, and 15-40% moisture; the improvement comprising:

- an enzyme in the pet food product consisting essentially of alpha-amylase, wherein the alpha-amylase is in an active condition and wherein the active condition alpha-amylase is in the pet food in sufficient quantity to maintain the soft texture of the pet food product during subsequent storage.

2. The improved pet food product according to claim 1 wherein the improved pet food product has a pH in the range 2-10.

3. The improved pet food product according to claim 1 wherein the alpha-amylase is present in the improved pet food product in an amount from a trace to 5,000 SKB units of alpha-amylase for each kilogram of starch.

4. In a method for making a soft, semi-moist pet food by the steps of admixing a dough having at least one amylaceous ingredient, at least one proteinaceous ingredient selected from animal, vegetable or cereal sources, sufficient water to provide a moisture content in the pet food product in the range 15-40% and at least one water soluble solute selected from sugars and polyols; and extrusion cooking the dough at a temperature in the range 75°-110° C. inclusive for a time period in the range 15 seconds to 5 minutes inclusive, the improvement comprising:

- (a) adding to the dough an enzyme ingredient consisting essentially of a heat-stable alpha-amylase enzyme;
- (b) extrusion cooking the dough containing alpha-amylase at a temperature and for a time period sufficiently mild that at least some alpha-amylase remains active in the extrusion cooked pet food product; and
- (c) storing the pet food product containing the active alpha-amylase.

5. The improved process according to claim 4 wherein from a trace of 5000 SKB units of alpha-amylase for each kilogram of amylaceous ingredient is added to the dough.

* * * * *

EVIDENCE APPENDIX

EXHIBIT A: Office Action dated February 22, 2006

EXHIBIT B: *Kobata* (U.S. Patent No. 4,540,585), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT C: *Nakagawa* (U.S. Patent 5,835,911), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT D: *Chen et al.* (U.S. 5,797,016), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT E: *Valentine* (U.S. 6,018,654), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT F: *Pepe et al.* (U.S. Patent 5,742,668), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT G: *Shear* (U.S. Patent 4,977,594), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT h: *Shah* (U.S. Patent 6,029,065), cited by the Examiner in the Office Action dated February 22, 2006

EXHIBIT C



US005835911A

United States Patent [19][11] **Patent Number:** **5,835,911****Nakagawa et al.**[45] **Date of Patent:** **Nov. 10, 1998****[54] SOFTWARE DISTRIBUTION AND MAINTENANCE SYSTEM AND METHOD**

5,390,314	2/1995	Swanson	395/500
5,526,257	6/1996	Lerner	364/401
5,544,320	8/1996	Konrad	395/200.09

[75] Inventors: **Toru Nakagawa; Yuji Takada**, both of Kawasaki, Japan

Primary Examiner—Thomas G. Black
Assistant Examiner—Cheryl Lewis
Attorney, Agent, or Firm—Staas & Halsey

[73] Assignee: **Fujitsu Limited**, Kanagawa, Japan**[57] ABSTRACT****[21] Appl. No.:** **517,133****[22] Filed:** **Aug. 21, 1995****Related U.S. Application Data****[63] Continuation-in-part of Ser. No. 385,460, Feb. 8, 1995, abandoned.****[30] Foreign Application Priority Data**

Feb. 8, 1995	[JP]	Japan	06-014706
Feb. 8, 1995	[JP]	Japan	06-014710
Aug. 17, 1995	[JP]	Japan	7-233422

[51] Int. Cl.⁶ **G06F 17/30****[52] U.S. Cl.** **707/203; 707/3; 707/10; 395/200.33; 395/200.51****[58] Field of Search** **395/608, 610, 395/619, 500, 200.09, 700, 200.33, 200.51; 364/401, 300; 707/203, 10, 3****[56] References Cited****U.S. PATENT DOCUMENTS**

4,558,413	12/1985	Schmidt et al.	364/300
5,247,683	9/1993	Holmes et al.	395/700

A number of sets of software may be systematically distributed and maintained via a network connecting many vendors and users of client/server software. A client program in a user computer detects when software subject to maintenance is activated and transmits an inquiry over the network to the software vendor's computer for information on the current version of the software. The server program compares data in the inquiry with data relating to the latest version of the software and returns update instruction information and updated software if appropriate. The client program automatically updates the software to the latest version according to the update instruction information when it is received. The client program can also send inquiries at predetermined times, or in response to a user command. The inquiry can include a request for purchase information in which case the server checks qualifications of the user, processes the inquiry according to vendor management data and returns the requested software, if appropriate. Other inquiries can also be made in response to user commands or automatically, e.g., to obtain information on the most recent version and transmission of data from client to server in response to an abnormal termination of client software.

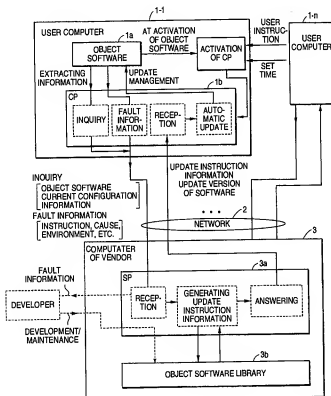
94 Claims, 24 Drawing Sheets

FIG. 1

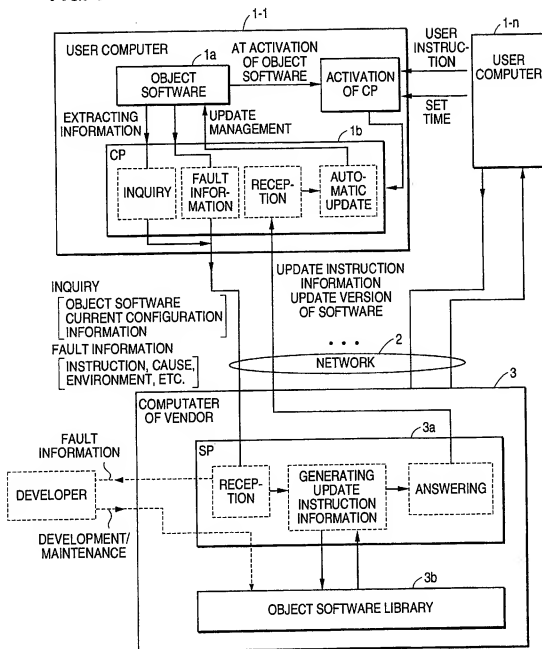


FIG. 2

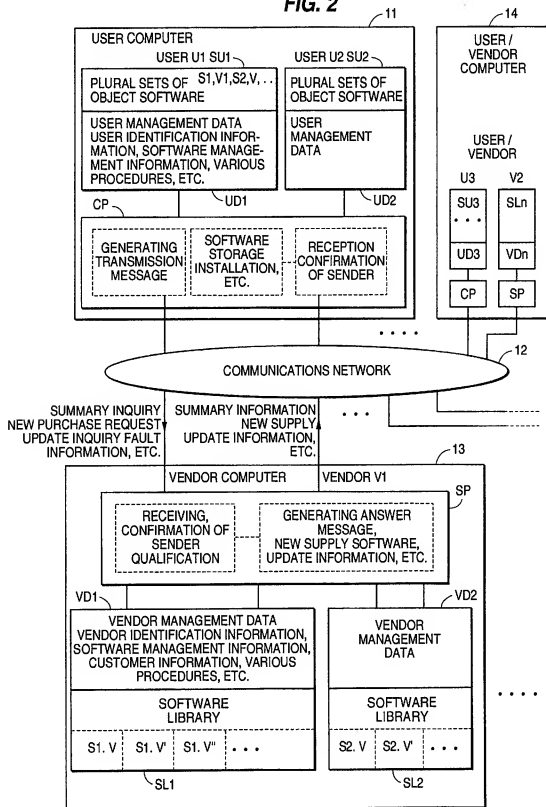


FIG. 3

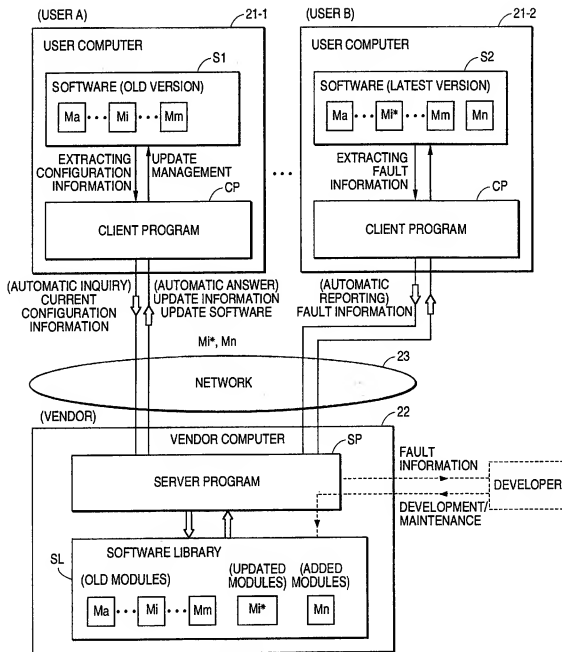


FIG. 4

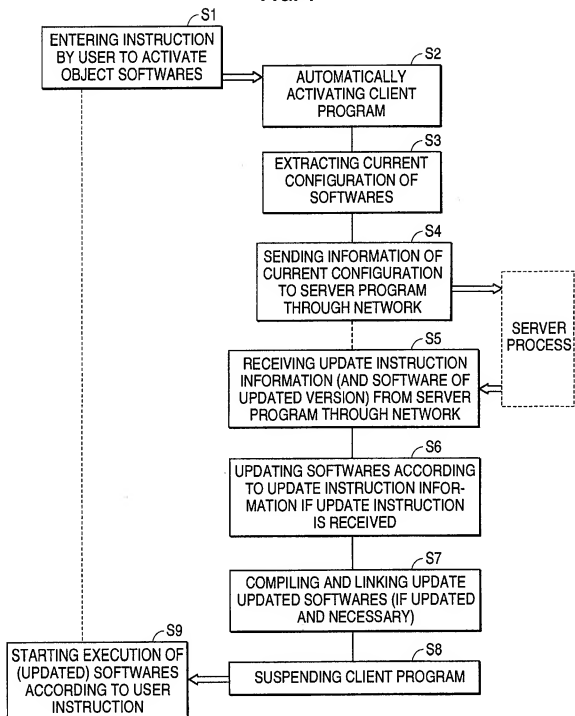


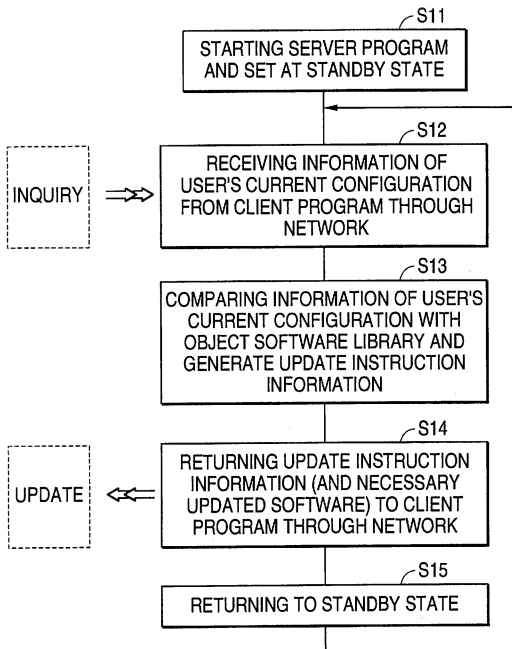
FIG. 5

FIG. 6

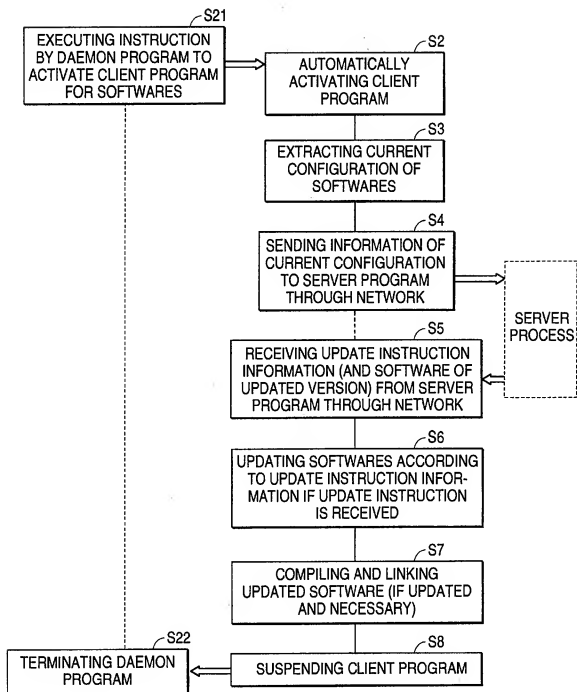


FIG. 7

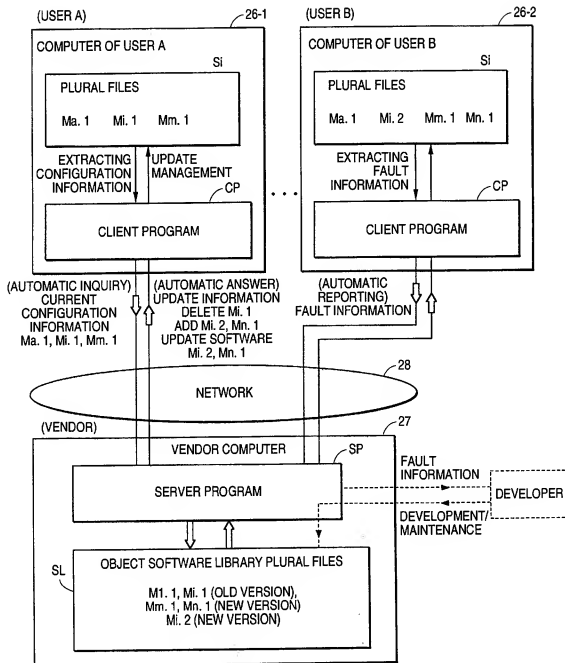


FIG. 8

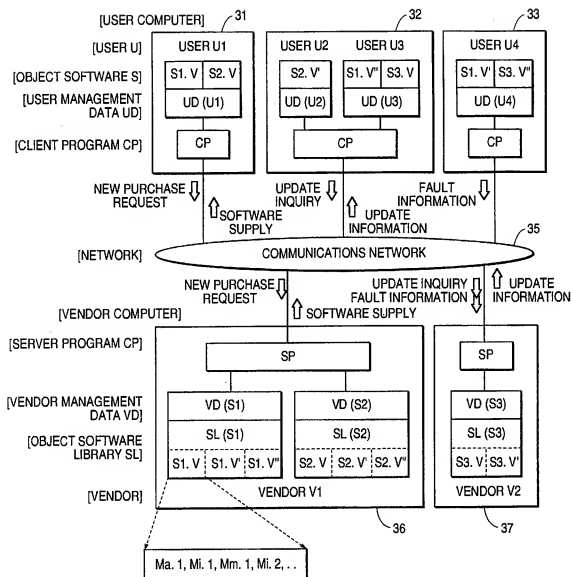


FIG. 9

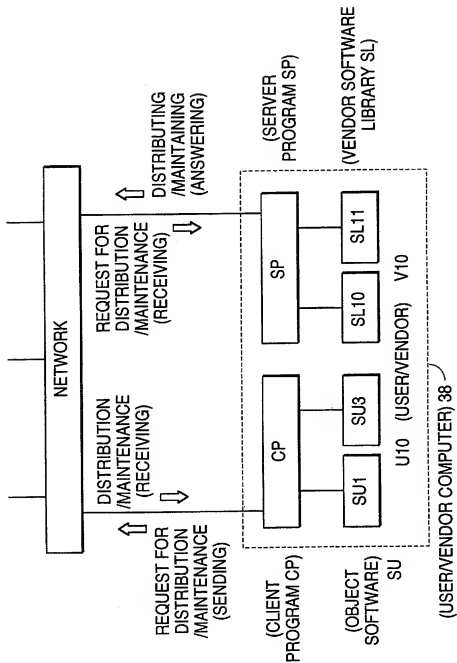


FIG. 10

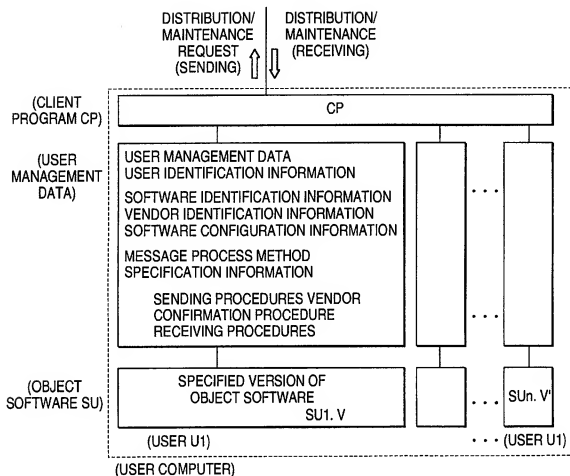


FIG. 11

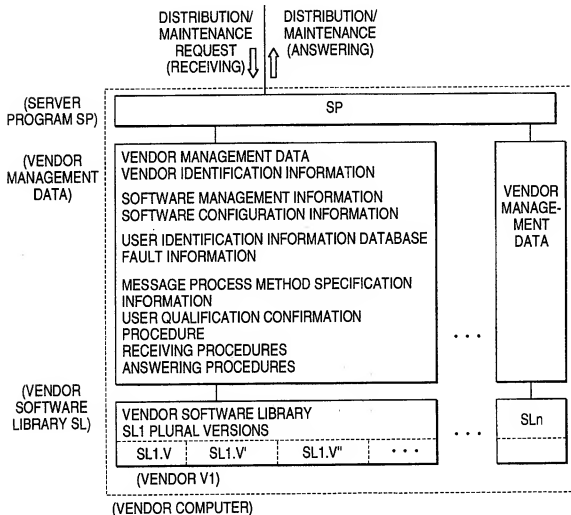


FIG. 12

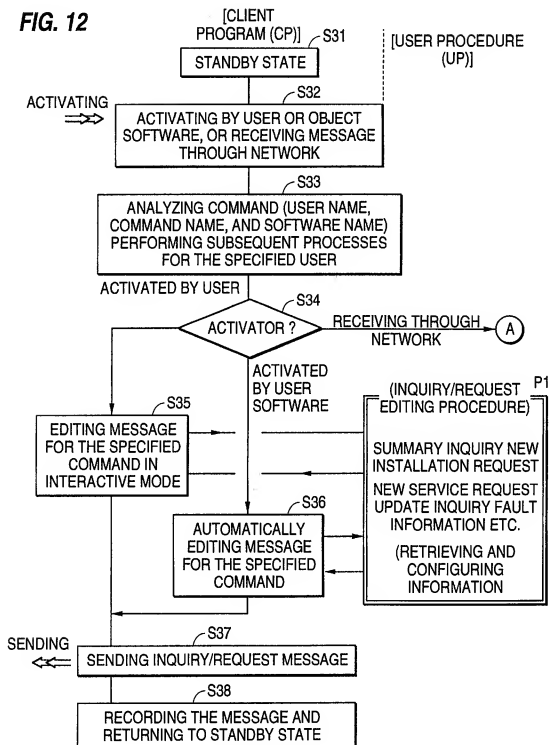


FIG. 13

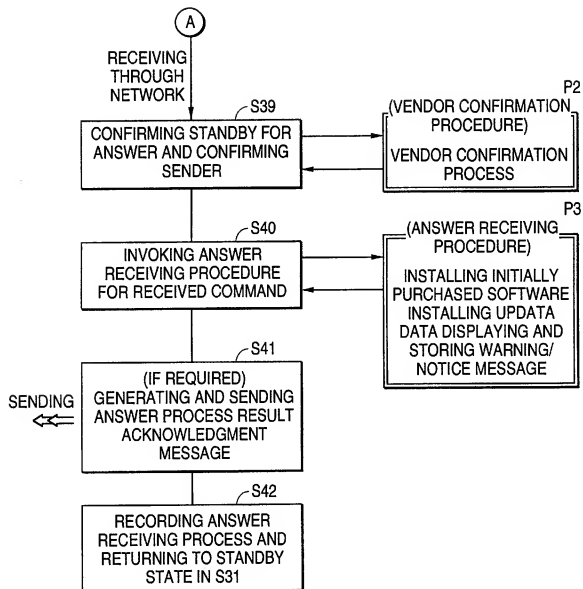


FIG. 14

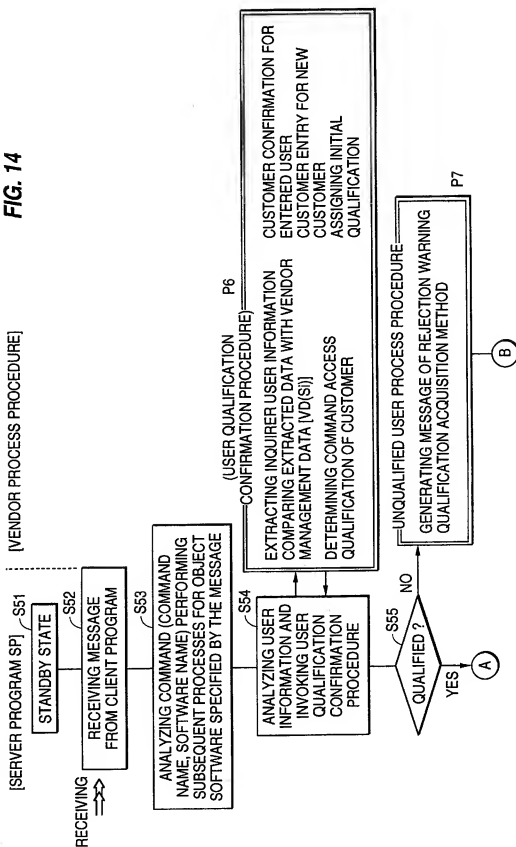


FIG. 15

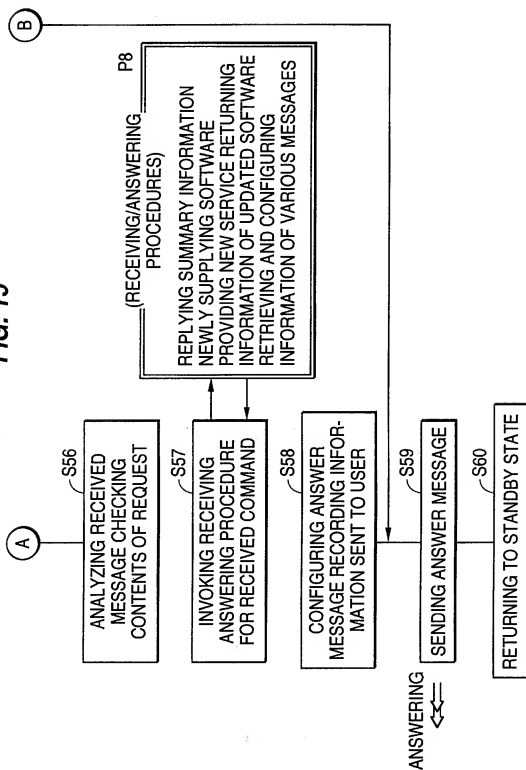


FIG. 16

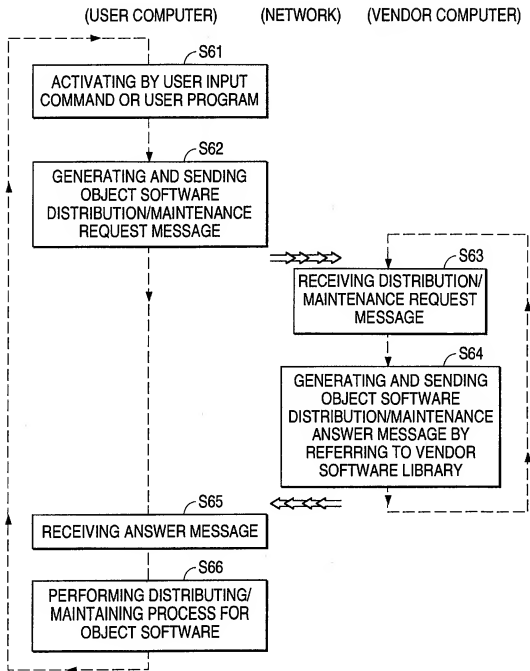


FIG. 17

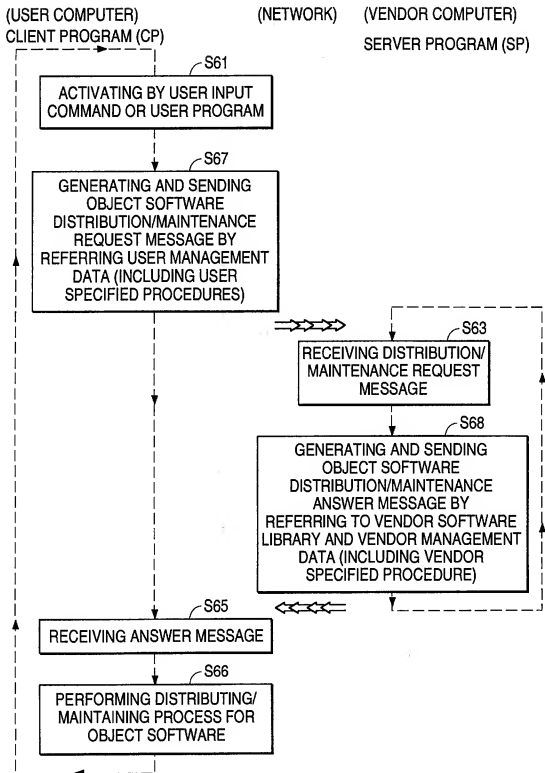


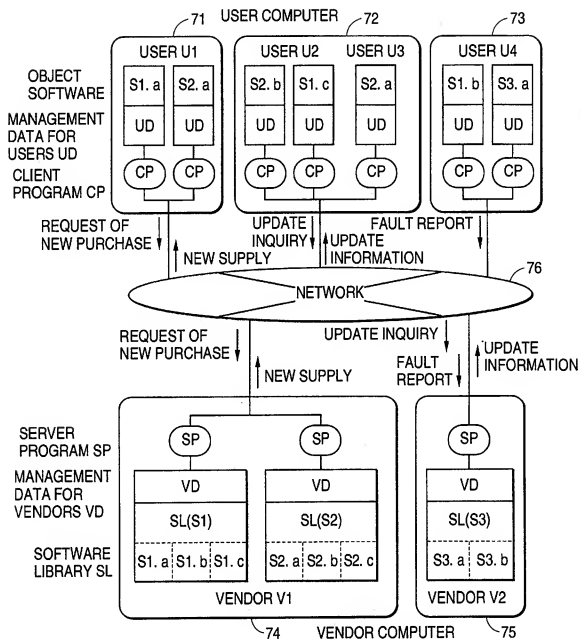
FIG. 18

FIG. 19

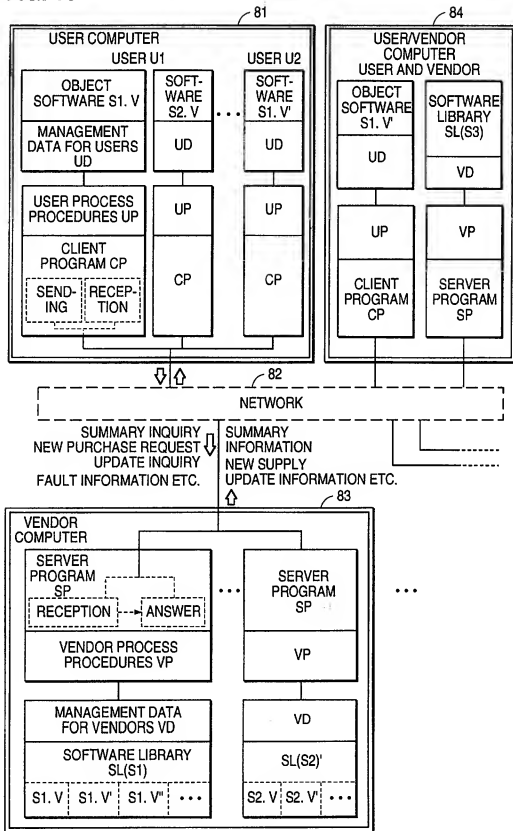


FIG. 20

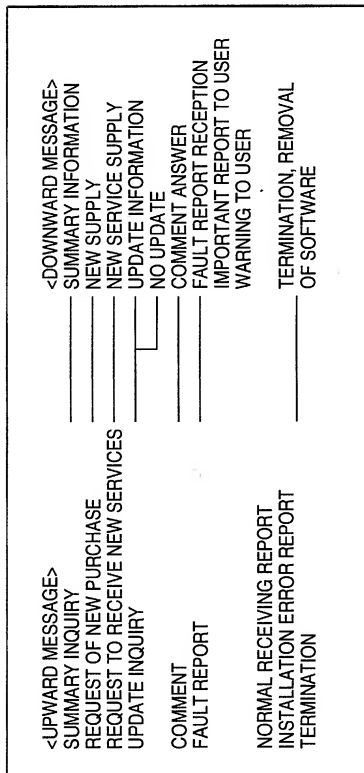


FIG. 21

CLASSIFICATION	COMMAND NAME	COMMAND MEANINGS
UPWARD MESSAGE	Inform WantNew WantRenew CheckUpdate BugReport Comment InitAck InitITrouble FinishUse	INQUIRY ABOUT SUMMARY INFORMATION, NEW SERVICES, PURCHASE METHOD ETC. REQUEST OF NEW PURCHASE REQUEST OF ADDITIONAL PURCHASE OF NEW SERVICES REQUEST TO SEND AN UPDATE INQUIRY AND UPDATED SOFTWARE INFORMATION AUTOMATIC REPORT OF FAULT INFORMATION DURING EXECUTION COMMENT, COMPLAINT, OPINION, QUESTION FROM USER REPORT OF PROPER SOFTWARE ACQUISITION/INSTALLATION ERROR REPORT WHEN INSTALLING PURCHASED SOFTWARE REPORT OF TERMINATION
DOWNWARD MESSAGE	ReplyInfo NewInstall Renew NoUpdate Update Answer BugAck Notice Warning Erase	ANSWERS SUCH AS SUMMARY INFORMATION, NEW SERVICES, PURCHASE METHOD ETC. NEW SOFTWARE SUPPLY (ADDITIONAL) SUPPLY OF NEW SOFTWARE SERVICES ANSWER OF NO UPDATE SENDING UPDATED SOFTWARE INFORMATION ANSWER TO USER'S COMMENT, COMPLAINT, OPINION, AND QUESTION ANSWER OF RECEIVING AUTOMATIC REPORT OF FAULT INFORMATION IMPORTANT REPORT TO USER WARNING TO USER REMOVAL OF PROGRAM SENT BY USER AT TERMINATION

FIG. 22

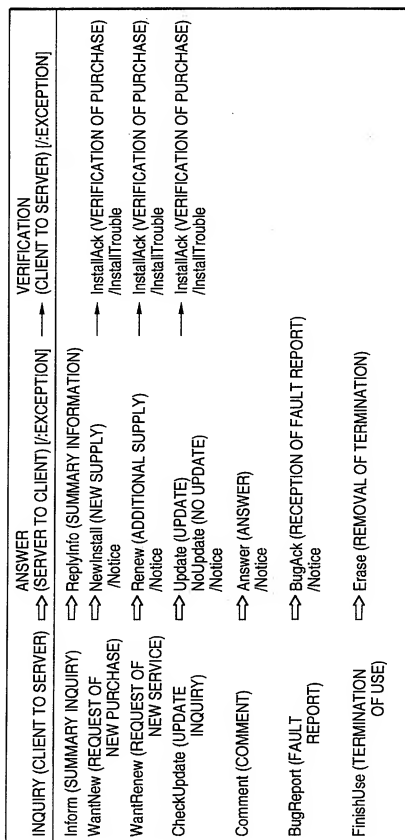


FIG. 23

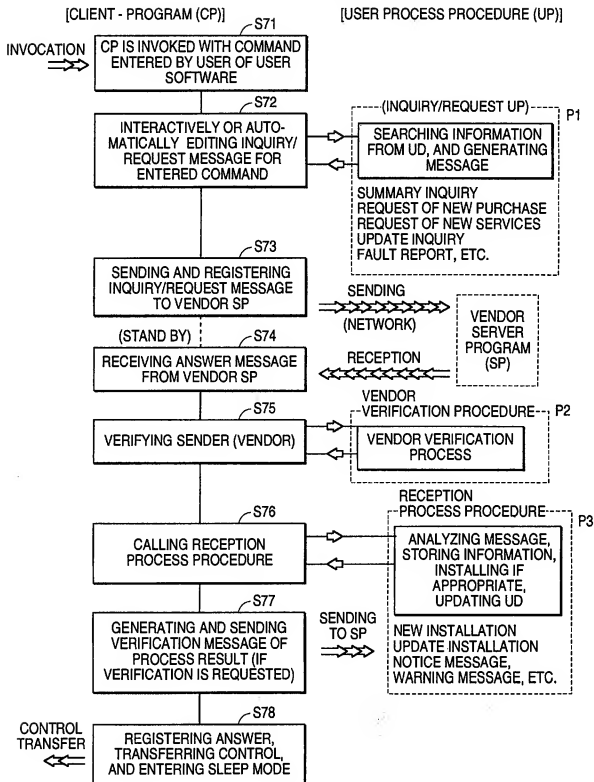
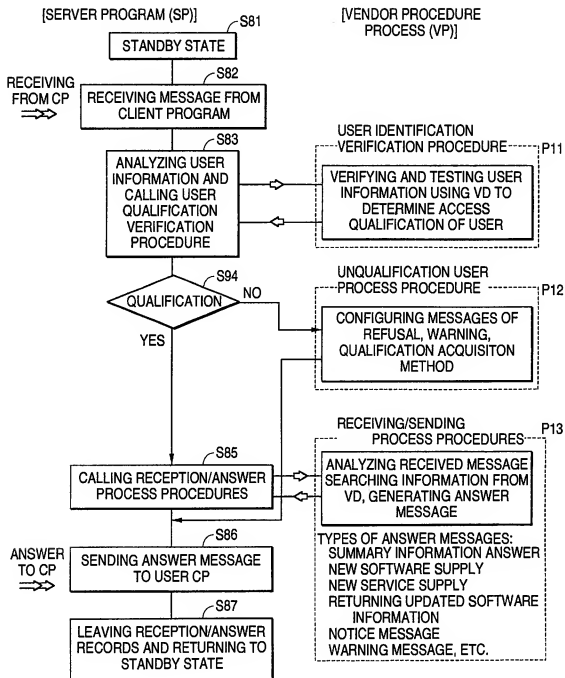


FIG. 24



1

SOFTWARE DISTRIBUTION AND MAINTENANCE SYSTEM AND METHOD

CROSS REFERENCE TO RELATED APPLICATION

This application is a continuation-in-part application of U.S. patent application Ser. No. 08/385,460 filed on Feb. 8, 1995, now abandoned.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a software distribution and maintenance system and method with which a software vendor can provide a number of users with software over a network, and update and maintain the software at requests of the users, and with which the users obtain a lot of software from plural software vendors over the network and can use the latest versions of the software over the network.

2. Description of the Related Art

The well-known prior art to distribute and obtain software update information is as follows.

(a) Method of Distributing Software using Portable Media

To sell and distribute computer software to users, the software is normally stored in portable media such as magnetic tapes, floppy disks, etc.

In this case, it is necessary to provide additional services such as correcting bugs, adding functions, and supplying new versions.

If the additional services are to be installed in the users' computer systems, the vendors have to visit the users' offices, or the users must install necessary services by themselves.

(b) Method of Distributing Software over a Network from a Vendor

Recently, software is transmitted over a communications network to users. Information required to correct bugs, add functions, and supply new versions can also be provided for users from vendors over a network. If a vendor is informed of a user address, then the vendor can decide to transmit necessary information to the user.

When computer software is actually updated in user computer, the software should be re-installed by the user according to the software update information and instructions from the vendor of the software.

(c) Method of Users' Obtaining Software over a Network

Recently, a user can obtain a software over a network, when required, by accessing a software library which is stored in a vendor (or an agent) computer file (download). Likewise obtained at a request of a user is the information about correcting bugs, adding functions, and supplying new versions. The vendor only has to manage the library. The obtained software is installed, updated, and managed in the user computer at the user's responsibility.

In this case, the software used to obtain a necessary software over a network is quite separate from the necessary software, and therefore is not used to update the necessary software. The user has to decide which software/information to be obtained and is responsible for the access operations.

(d) Linking and Processing Distributed Software over a Network

Software should be appropriately converted in an executable format normally by compiling and linking source programs. Usually, the source programs are stored in a user computer and then compiled and linked when necessary. However, the source programs can be stored in another computer (that is, the vendor computer). In this case, the

2

user accesses the vendor computer from the user computer over a network, then either (1) compiles and links the source programs in the vendor computer to download the resultant programs or (2) downloads the source programs and then compiles and links them in the user computer, thereby converting the software into an executable format.

This method fundamentally belongs to the above described method (c), but is an easier method of executing software.

In this method, the user recognizes that the source programs are stored in another computer, and then clearly specifies necessary modules and files (for example, specifies the name of a file with its version number) to transfer and link the programs. The entire process is performed at the user's responsibility.

Described below are the prior arts partially similar to the above described technology but different in purpose, object, and method.

(e) Automatic Download

This method is used for an electronic notice board. In this method, a large amount of news information is stored in a central computer, etc., and a terminal computer can access the central computer through the network periodically at intervals and automatically read new information in a specified category if newly stored (download).

In this case, the central computer is regarded as a vendor computer, and the terminal computer is regarded as a user computer. Object information is news, software (so-called freeware), etc.

The software which is automatically downloaded is stored as ordinary data, and does not directly update or manage the software currently used in the user computer. That is, the above described automatic download system does not manage the software currently used by a user.

(f) Mirroring

A mirroring process is used to access new/updated data and obtain the latest information among a plurality of central computers, etc. which store and provide a large amount of document data, etc.

A central computer is a primary storage center of information of a specific category. Stored information is copied to other plural central computers (secondary storage centers) for disclosing the stored information.

To attain this, the secondary storage centers periodically access the primary storage center, and detect and read newly stored/updated information only. This method is different from the above described automatic downloading method (e) in obtaining information in plural categories by switching for each category the functions of the primary storage center and the secondary storage centers among the central computers in a network.

In this method, the information exchanged among the central computers (including software) is handled as ordinary data. That is, the software currently used by a user is not directly updated or managed through mirroring.

(g) Remote Maintenance over a Network

If a fault occurs in a user computer, a software (or hardware) vendor directly operates user software through a network at a user's request (or through automatic communications over the network) to detect the reason for the fault and recover from the fault.

This process is normally performed as a service individually provided by a vendor engineer to recover from a software fault.

That is, the process is different from an automatic supply of maintenance and update services by automatic software for a number of users.

(b) Client/Server Method

The client/server method, in which information is communicated between the client and server software and a client operates server software, is well known and has been used to realize the above listed technologies (b)-(f). No systems, however, have been developed so far with this method to automatically update software already distributed to a user, by simply updating the software in the vendor computer.

Generally, computer software has been designed as an advanced, complicated, and large-scale product with wide variation, and is used by the increasing number of users in various fields.

Under such conditions, optimally maintaining, managing, updating, and improving user computer software is a big consuming job to vendors and users.

Nevertheless, according to the above listed conventional methods, the software update information is distributed individually by a vendor to a user as in the cases of (a) and (b), or is obtained by the user as in the cases of (c) and (d). It is determined individually either by a vendor or by a user which version or part of software is transmitted and obtained, and the software is manually installed or re-installed. These manual operations cause many inconveniences and problems.

From a viewpoint of vendors, since software is supplied to a large number of users on a variety of time schedule and is independently managed by users, it brings forth various problems when the software should be optimally maintained and managed.

That is, the following problems are caused in the above listed prior art technologies.

<Problem 1> Since a great number of users obtain and use software with different configurations according to different schedules, it is very difficult for vendors to correctly recognize the configuration of the software of each user.

<Problem 2> Although a bug is detected and the vendor software is corrected in a vendor computer, the software in a user computer cannot be corrected immediately. Therefore, the user faces inconvenience until the incomplete software is corrected in a user computer.

<Problem 3> It takes a long time to provide all users with new services such as extended functions, new versions, etc. <Problem 4> Since actual users of software cannot be sufficiently recognized by a vendor in many cases, the vendor cannot send necessary information promptly to the users. This problem grows as the users increase in number and in variation, especially for commercially distributed software.

<Problem 5> Since a large cost is required when a vendor has to visit a user to install software, it is practically impossible to install the software in many cases. Whereas not all services provided by the vendor for the software can be realized when the user is made responsible to install the software.

From a viewpoint of users, a quick and proper service cannot be given by the vendor, and a time-consuming process is required onto the user to install, update, or manage the software. Thus, the software cannot be sufficiently utilized.

<Problem 6> Sufficient knowledge and time consuming work are required to optimally install software depending on the hardware/software environment of each user and to manage the versions of the installed software, thereby permitting only skilled engineer to perform the required processes.

<Problem 7> If software contains bugs, an object job cannot be performed properly or smoothly in a user computer, and

the user must obtain a bug-corrected version from the vendor and install it with taking a long time for the process.

<Problem 8> Even when new functions are added or new versions are developed by the vendor, they are not significantly until the users can obtain and re-install them successfully. Obviously, long processes and time are required to obtain and re-install them.

<Problem 9> Users are often ignorant of the information of bug correction, function addition, new version development, etc. pertaining their software and are using the old software as being incomplete.

<Problem 10> It is troublesome for a user to frequently receive information from the software vendor about bug correction, additional functions, etc. and to have to install additional technologies.

Summing up the above listed problems, they are caused by manual processes performed by software vendors or users in distributing/obtaining software update information (and updated software) and updating the software in the users' computers (re-installing the software).

A comparatively simple system to solve these problems can be a system of automatically updating an object software in a computer of each user to the latest version by use of a client/server method. A single object program is managed by client programs in a number of user computers and is supported by a server program which manages the software library of the object software in a vendor computer.

However, such a system cannot be successfully designed without solving the following various problems of wide-purpose and wide-variation software of late.

<Problem 11> A number of users refer to wide variation. For example, (1) various computers, (2) various software environments in user computers, (3) various electronic skill levels of users, (4) various skill levels and purposes of use for an object software, (5) various uses of user computers (for office use, personal use, etc.), and the like.

Considering the above listed variation, client programs must have sufficient variety in the process to manage software in user computers. Software vendors and server programs in vendor computers should properly recognize the above listed variation at users'.

<Problem 12> The variation pertaining to a number of software affects the distribution and management of the software.

There are various types of software which differ in distribution and usage. For example, (1) product software, published common-use software, and intra-office software, (2) embedded software, basic software (OS), and application software, (3) operating system, online software, and batch run software, (4) load module provided, object module provided, and source provided software. Each type of software should be processed appropriately.

<Problem 13> Even a single user has different levels and phases in using a single object software.

For example, there are different levels or phases of usage as follows: (1) a phase having no purchase right, and a phase having a purchase right, (2) a phase of without initial installation and a phase complete initial installation, (3) beginner user phase, stable user phase, and advanced user phase, (4) conventional version phase, version switching phase, new version phase, (5) a phase of stable use of conventional environment, a phase of trial use of new environment, and new environment phase, etc.

These phase differences should be sufficiently considered for each user to obtain software services properly.

<Problem 14> A single user usually adopts plural sets of object software provided by different vendors. It is not

convenient for users that plural sets of object software are managed by different distribution/maintenance systems provided by different vendors. A more systematic and convenient system is required by users.

Summarizing the above problems 11 through 14, a unified system and method for distributing and maintaining software are required where a number of various software users and a number of vendors can systematically manage the distribution and maintenance of plural sets of various object software.

SUMMARY OF THE INVENTION

Object of the Invention

The present invention has been developed to solve the above described problems of the prior art technologies.

The first object of the present invention is to provide a software distribution/maintenance system and method over a network so that a number of various software vendors and users can systematically manage plural sets of various object software.

The second object of the present invention is to provide a software distribution/maintenance system and method over a network so that users can quickly and properly obtain and use the software developed and updated by the software vendor.

The third object of the present invention is to provide a software distribution/maintenance system and method over a network so that computer users connected to the network can quickly and properly obtain software used by another user in the network.

The fourth object of the present invention is to provide a software distribution/maintenance system and method over a network so that users can make requests and inquiries to software suppliers in various formats, for example, inquiries of the outline of software, requests for initial purchase of software, inquiries of update, request for new services, etc.

The fifth object of the present invention is to provide a software distribution/maintenance system and method over a network in order to distribute and maintain various types of software such as product software, shareware, freeware, scientific prototype software, intra-office software, etc.

The sixth object of the present invention is to provide a software distribution/maintenance system and method over a network so that even an unskilled user can obtain a new software or an updated software in an immediately operable form.

The seventh object of the present invention is to provide a software distribution/maintenance system and method over a network so that a software vendor can immediately detect an occurrence of a software fault in order to quickly correct the fault.

The eighth object of the present invention is to provide a software distribution/maintenance system and method over a network which allows to unify a program of managing the software of user computers and to unify a program of managing each software library of vendor computers.

The ninth object of the present invention is to provide a software distribution/maintenance system and method over a network which allows flexible message communication between users and vendors by introducing procedure definition of methods of processing the messages.

The tenth object of the present invention is to provide a software distribution/maintenance system and method over a network which allows secured message communication

between users and vendors by introducing procedure definition of methods of confirming a vendor and determining the qualification of a user.

The eleventh object of the present invention is to provide software on a worldwide scale, or build a standardized technical framework suitable for a form of software distribution without making the designs of the client program CP and the server program SP complicated. That is, the present invention is intended to provide a software distribution and maintenance system and method, which uses a number of and a variety of pieces of software as objects and can be utilized by a number of software users joined on a worldwide scale, comprises a capability to rapidly and properly acquire and use software created and updated by software vendor.

A feature of the present invention resides in a software distribution/maintenance system having a plurality of user computers and a vendor computer connected to the plurality of user computers through a network to manage and automatically update over the network a set of object software sent and stored in the user computers from the vendor computer through the network, comprising: first process means in the user computers; second process means in the vendor computer; and object software library in the vendor computer, wherein said first process means sends current configuration information of the object software stored in the user computers to said second process means of the vendor computer through the network to inquire the latest configuration, receives an answer from said second process means, and updates the object software stored in the user computers according to an instruction in a received answer; and

said second process means receives an inquiry from said first process means in the user computers, generates update instruction information for the object software to match a configuration of the object software in the user computers with the configuration of an updated version in said object software library stored in the vendor computer, and returns through the network to the first process means the update instruction information and the software of the updated version.

Another feature of the present invention resides in a software distribution/maintenance system having a plurality of user computers and a vendor computer to manage and automatically update object software provided for the plurality of user computers from the vendor computer through a network, comprising: first process means in each of the user computers; and

network means for connecting the user computers to the vendor computers; wherein said first process means sends current configuration information of the object software stored in the user computers to the vendor computer through the network to inquire the latest configuration, receives an answer from the vendor computer, and updates the object software stored in the user computers according to an instruction in a received answer.

A further feature of the present invention resides in a software distribution and maintenance system using a network in which a number of users U1, U2, . . . using a number of types of object software to be distributed, managed, and maintained, and a number of software vendors V1, V2, . . . supplying the object software manage the distribution and maintenance of the object software over a computer network, comprising: one or more first process means CPs installed in each of user computers, that manage object software groups S1, S2, . . . to be used by one of more users

U1, U2, . . . individually for every object software and for every user; one or more second process means SPs installed in either of software vendor computers, that gives services to vendor software libraries SL (S1), SL (S2), . . . for each of the software libraries; a network that connects the first process means CP installed in the user computer to the second process means SP, based on standardized communications protocols; and wherein said first process means CP has a capability that perform distribution and/or maintenance of the object software by sending a message that requests to distribute and/or maintain the object software for one piece of object software over the network, according to instructions given by the users U1, U2, . . . or a user-defined program, receiving an answer message from the second process means SP, and processing it depending on contents of the answer message and settings made by the users U1, U2, . . . ; and said second process means SP has a capability to receive the message from an arbitrary first process means CP, to reference the software libraries SL(S1), SL(S2), . . . managed by the vendors V1, V2, . . . depending on the contents of a received message and settings made by the vendors V1, V2, . . . for the object software specified with the message, to generate an answer message to answer the request of distribution and/or maintenance of the object software, and to send the answer message to said first process means CP of the sender of the corresponding message.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows the principle of the first embodiment of the present invention;

FIG. 2 shows the principle of the second embodiment of the present invention;

FIG. 3 shows the system configuration according to the first embodiment;

FIG. 4 shows the process performed by a user computer according to the first embodiment;

FIG. 5 shows the process performed by a server program of a vendor computer according to the first embodiment;

FIG. 6 shows the process for periodical update in a user computer according to the first embodiment;

FIG. 7 shows a practical example of the first embodiment;

FIG. 8 shows the system configuration according to the second embodiment;

FIG. 9 shows the computer as a vendor-and-user computer according to the second embodiment;

FIG. 10 shows the contents of user management data in a user computer according to the second embodiment;

FIG. 11 shows the contents of the vendor management data in a vendor computer according to the second embodiment.

FIG. 12 shows the entire configuration of the process performed by a client program according to the second embodiment;

FIG. 13 shows the entire configuration (continued) of the process performed by a client program according to the second embodiment;

FIG. 14 shows the entire configuration of the process performed by a server program according to the second embodiment;

FIG. 15 shows the entire configuration (continued) of the process performed by a server program according to the second embodiment;

FIG. 16 is a flowchart showing the entire process of the software distribution/maintenance method according to the second embodiment;

FIG. 17 is a flowchart showing the entire process performed by the software distribution/maintenance method using user management data and vendor management data;

FIG. 18 shows a block diagram of a principle structure according to the third embodiment of the present invention;

FIG. 19 shows a block diagram of the third embodiment shown in FIG. 18 of a software distribution and maintenance system utilizing a network according to the present invention;

FIG. 20 designates an upward message and a downward message according to the third embodiment;

FIG. 21 depicts an example of a processing command;

FIG. 22 shows the correspondence relationship between various messages;

FIG. 23 shows a flowchart of an entire process performed by a client program CP according to the third embodiment; and

FIG. 24 shows a flowchart of an entire process performed by a server program SP according to the third embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

First Preferred Embodiment for Distributing/Maintaining an Object Software

According to the first embodiment of the present invention, a client program is stored in each user computer to manage an object software to be distributed and maintained, and a server program is stored in a vendor computer to manage an object software library.

The client program sends configuration information indicating the modules forming the object software in the user computer to the server program of the vendor computer through the network, inquires whether or not the configuration of the present object software matches the configuration of the latest configuration, receives an answer from the server program, and updates the object software according to the answer.

On the other hand, the server program receives an inquiry about the configuration of software from any client program in a plurality of user computers, generates update instruction information for the object software so that the configuration of the user software matches the latest version of the object software in the vendor computer, and answers the inquiring client program, by providing the update instruction information and the software of the latest version over the network.

Second Preferred Embodiment for Distributing/Maintaining Multiple Object Software of Multiple Vendors

The above described configuration is used in an automatic remote software update system in which a single object software is used as a principle. Furthermore, according to the second embodiment, each user computer is usually provided with plural sets of object software, and each vendor computer is usually provided with a plurality of software libraries.

According to the second embodiment, each user computer stores a client program for managing plural sets of software used by a plurality of users, and each vendor computer stores a server program for managing a plurality of software libraries provided by the vendor.

The client program in each user computer sends according to the instruction of the user over a network a message requesting to distribute/maintain a single object software to the server program in the vendor computer from which the software is supplied, and then performs processes to distribute and maintain the object software according to the answer

message from the server program and the answer instruction predetermined by the user.

In response to this, when the server program receives a message from the client program in any user computer, it refers to the software library of the object software specified by the message according to the received message and the received instruction predetermined by the vendor, generates an answer message to the distribution/maintenance request message, and returns the answer message to the client program over the network.

According to the present invention, the server program in the vendor computer returns the answer message in response to the software distribution/maintenance request message from the client program, thereby automatically distributing/maintaining the object software as described above.

Explanation of the Principle

Explanation of the Principle of the First Embodiment

Described first is the principle of the first embodiment of the present invention.

FIG. 1 shows the principle of the first embodiment. In FIG. 1, user computers 1-1, . . . , 1-n are connected to a vendor computer 3. Each of the user computers 1-1, . . . , 1-n is equipped with an object software 1a to be updated/managed and a first process unit 1b (which may be called a client program CP) for managing the configuration and operation of the object software 1a, for example. A user can perform a necessary job by operating the object software 1a.

3 is a vendor computer. The vendor computer 3 is provided with a second process unit 3a (which may be called a server program SP) for managing the object software 1a stored in the user computers 1-1, . . . , 1-n, and an object software library 3b.

To solve the above described problems, a software distribution/maintenance system according to the present invention comprises, as shown in FIG. 1, a plurality of the user computers 1-1, . . . , 1-n and the vendor computer 3 connected to the plurality of user computers 1-1, . . . , 1-n through the network 2, and manages and automatically updates over a network the object software 1a stored in the user computers 1-1, . . . , 1-n from the vendor computer 3 through the network 2.

In this system, the user computers 1-1, . . . , 1-n comprise the first process units 1b, and the vendor computer 3 comprises the second process unit 3a and the object software library 3b.

The first process unit 1b sends the current configuration information of the object software 1a stored in the user computers 1-1, . . . , 1-n to the second process unit 3a of the vendor computer 3 through the network 2 to inquire the latest configuration, receives an answer from the second process unit 3a, and updates the object software 1a stored in the user computers 1-1, . . . , 1-n according to the instruction in the received answer. The second process unit 3a receives the inquiry from any of the first process units 1b stored in the user computers 1-1, . . . , 1-n, generates update instruction information for the object software 1a to match the configuration of the object software 1a in the user computers 1-1, . . . , 1-n with the configuration of the updated version in the object software library 3b stored in the vendor computer 3, and returns through the network 2 to the inquiring first process unit 1b the update instruction information and the software of the updated version.

Furthermore, according to the present invention, if the operation of the object software 1a abnormally terminates in the user computers 1-1, . . . , 1-n, the first process units 1b in the user computers 1-1, . . . , 1-n automatically inform over

the network 2 the vendor computer 3 of an abnormal termination and its state.

The first process unit 1b according to the present invention automatically informs the vendor computer 3 of the abnormal termination, the instruction causing the abnormal termination and the reason for the abnormal termination, a series of instructions which invoked the instruction causing the abnormal termination, and the environment of the software/hardware used when the abnormal termination is detected.

The first process unit 1b can be designed to automatically inform over the network 2 the vendor computer 3 of the fact and state of the termination of an abnormal condition by general-purpose electronic mail.

According to the present invention, the software distribution/maintenance method manages and automatically updates the object software 1a stored in the user computers 1-1, . . . , 1-n from the vendor computer 3 through network 2.

In this method, the first process units 1b in the user computers 1-1, . . . , 1-n send the current configuration information of the object software 1a to the second process unit 3a in the vendor computer 3 through the network 2 to inquire the latest configuration.

The second process unit 3a in the vendor computer 3 receives the above described inquiry, generates update instruction information for the object software 1a to match the configuration of the object software 1a in the user computers 1-1, . . . , 1-n with the configuration of the updated version in the object software library 3b stored in the vendor computer 3, and returns through the network 2 to the inquiring first process unit 1b the update instruction information and the software of the updated version.

The first process units 1b stored in the user computers 1-1, . . . , 1-n receive an answer from the second process unit 3a, update the object software 1a according to the update instruction information in the answer, and prepare for the compiling and linking of programs if necessary.

There can be plural ways of activating the first process unit 1b. Three ways of them are shown below:

When the object software 1a is activated in the user computers 1-1, . . . , 1-n, the first process unit 1b immediately sends the current configuration information of the object software 1a to the second process unit 3a in the vendor computer 3 through the network 2 to inquire the latest configuration, receives an answer from the second process unit 3a, updates the object software 1a according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary.

When the first process unit 1b is activated in the user computers 1-1, . . . , 1-n at a predetermined time, the first process unit 1b sends the current configuration information of the object software 1a to the second process unit 3a in the vendor computer 3 through the network 2 to inquire the latest configuration, receives an answer from the second process unit 3a, updates the object software 1a according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary. Thus, the object software 1a can be automatically updated.

When a user instructs the user computers 1-1, . . . , 1-n to activate the first process unit 1b, the first process unit 1b sends the current configuration information of the object software 1a to the second process unit 3a in the vendor computer 3 through the network 2 to inquire the latest configuration, receives an answer from the second process unit 3a, updates the object software 1a according to the

update instruction information in the answer, and prepares for the compiling and linking of programs if necessary. Thus, the object software 1a can be automatically updated.

The above described methods can be selectively used to activate the first process unit 1b.

Function of the First Embodiment of the Present Invention

In FIG. 1, if a user activates the object software 1a of the user computers 1-1, . . . , 1-n, then the first process unit 1b detects the activation and sends configuration information of the current version to the second process unit 3a in the vendor computer 3 through the network 2. When the second process unit 3a in the vendor computer 3 receives the information, it compares the current configuration with the configuration stored in the object software library 3b, and returns the instruction information for the update of the object software 1a of the user together with the update software. The user's first process unit 1b automatically updates the object software 1a using the received information.

The first process unit 1b can be activated at a predetermined time or at a user instruction as shown in FIG. 1, thereby updating the object software 1a.

Thus, the current configurations of the software in the computers of a number of users can be exactly recognized by the vendor, and the vendor can automatically provide the users with the latest version of the software and the latest information relating to the software.

Furthermore, the software can be automatically installed and the latest version of the software can also be automatically managed. Therefore, the vendor does not have to visit the user's office to install the software, and even beginners and unskilled users can correctly receive the latest version of the software.

In case that the operation of the object software 1a abnormally terminates in the user computers 1-1, . . . , 1-n, the first process units 1b in the user computers 1-1, . . . , 1-n automatically inform over the network 2 the vendor computer 3 of an abnormal termination and its state if the operation of the object software 1a abnormally terminates in the user computers 1-1, . . . , 1-n. Therefore, the user system automatically informs the vendor of the exact information of a fault of the software in the user computer when it occurs. The vendor corrects the software library according to the fault information. Then, the user can immediately use or operate the corrected software without a burden on the user.

The first process unit 1b automatically informs the vendor computer 3 of the abnormal termination, the instruction causing the abnormal termination and the reason for the abnormal termination, a series of instructions which invoked the instruction causing the abnormal termination, and the environment of the software/hardware used when the abnormal termination is detected. Accordingly, the vendor can efficiently correct the object software according to the above described detailed fault information. The first process unit 1b automatically informs the vendor computer 3 of an abnormal termination and state over the network 2 in the format of general-purpose electronic mail, thereby transmitting the abnormal termination information and the state in the common communications.

The first process unit 1b sends the current configuration information of the object software 1a to the second process unit 1a in the vendor computer 3 through the network 2 to inquire the latest configuration. The second process unit 3a in the vendor computer 3 receives the above described inquiry, generates update instruction information for the object software 1a to match the configuration of the object software 1a with the configuration of the updated version in

the object software library 3b, and returns through the network 2 to the inquiring first process unit 1b the update instruction information and the software of the updated version. The first process unit 1b receives an answer from the second process unit 3a, updates the object software 1a according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary.

When the object software 1a is activated, the first process unit 1b immediately sends the current configuration information of the object software 1a to the second process unit 3a in the vendor computer 3 through the network 2 to inquire the latest configuration, receives an answer from the second process unit 3a, updates the object software 1a according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary. Thus, the user can automatically update the object software 1a without considering the activation of the first process unit 1b.

When the first process unit 1b is activated at a predetermined time, the first process unit 1b sends the current configuration information of the object software 1a to the second process unit 3a to inquire the latest configuration, receives an answer from the second process unit 3a, updates the object software 1a according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary. Thus, the object software 1a can be automatically updated at night, for example, with the user released of being kept waiting in updating the object software 1a during the operation of the software. Furthermore, by the activation of the first process unit 1b at times predetermined by user's software, the object software 1a can be automatically updated periodically every dawn, every week, or every month without user's explicit operations.

When the user explicitly activates the first process unit 1b, it immediately sends the current configuration information of the object software 1a to the second process unit 3a in the vendor computer 3 through the network 2 to inquire the latest configuration, receives an answer from the second process unit 3a, updates the object software 1a according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary. Thus, the object software can be automatically updated by the user's initiation at any time when necessary.

The above described methods can be selectively used to update the object software 1a.

Explanation of the Principle of the Second Embodiment

Described below is the principle of the second embodiment. According to the second embodiment, unlike the first embodiment, each user computer normally comprises plural sets of software which are used by each user and provided by a plurality of vendors, and a plurality of vendor computers store a plurality of software libraries.

General Structure of the System

FIG. 2 shows the principle of the second embodiment. In FIG. 2, a user computer 11 is used by a plurality of users U1, U2, . . . (normally represented as user Uj). The users are provided with object software SU1, SU2, . . . (normally represented as object software SUj) to be managed by the present invention, user management data UD1, UD2, . . . (normally represented as management data UDj), and a third process unit CP, for example, a client program.

The third process unit CP sends to a vendor various messages from a user to the vendor such as software purchase requests, update inquiries, fault reports, etc.,

receives an answer from the vendor, updates user software, etc. The user management data UD1, UD2, ... contain user identification information, software management/configuration information, information (procedure) for regulating a processing method (method of installing software), etc. The information is used when various messages such as software purchase requests, update inquiries, fault reports, etc. are issued by the third process unit CP and when the user software is updated.

A vendor computer 13 is provided with software libraries SL1, SL2 (normally represented as SLi) of a vendor V1 (normally represented as a vendor Vk), vendor management data VD1, VD2 (normally represented by VDi), and a fourth process unit SP, for example, a server program.

The fourth process unit SP returns necessary information and software in response to various inquiries and requests from the users U1, U2.

The vendor management data VD1, VD2 contain data such as vendor identification information, software configuration information, customer management information, etc.; and regulatory information (procedure) for defining an answering method.

A user/vendor computer 14 is used by a user U3 and a vendor V2 (or a user can be a vendor) and they are individually provided with object software SU3, user management data UD3, the third process units CP, software libraries SLn, vendor management data VDi, and the fourth process units SP.

A network 12 connects the above mentioned computers 11, 13, and 14, and transmits messages communicated among them.

The configurations and versions of an object software are represented using the following terms and identification characters in this specification.

Each object software is referred to as object software Si. For example, S1 and S2 are different software. Since a single set of software Si has a number of version types depending on the type of computer, the specification of functions, etc., the version types are specified by respective characters such as Si.V, Si.V', Si.V'', etc. Furthermore, each version type is qualified with a version number for management of update order. For example, Si.V.1 indicates the version type and number of a set of software. The set of version type and number are referred to as a version for short.

The object software Si is supplied by a vendor (Vk) and managed in the software library SLi which stores software of all versions (Si.V.1, Si.V.2, ..., Si.V.1, ...) of all version types Si.V, Si.V', ..., Each object software Si normally comprises a number of modules M1, M2, ... (Commonly represented by Mm), and each of these modules is also assigned a new version number each time the module is updated. Thus, a module is represented by a module type and version number, for example, Mm.n. Generally, a software version Si.V.1 of software Si comprises a number of versions of modules (for example, M1.2, M2.1, M3.5, ...), and a specific module version can be shared by different software versions.

When users adopt object software Si, they normally desire to select the optimum version type Si.V from among a number of version types and then use the latest version Si.V.1 in the selected type.

Usually, a single user uses plural sets of software. Plural sets of object software SUj used by user Uj comprise a number of versions of software (for example, S1.V.1, S2.V.3, S3.V''.1, ...).

As described above, a vendor manages a number of versions for a single set of software, and a user manipulates

plural sets of software (supplied by different vendors), but usually a single version for each set. Thus, the second embodiment is based on the supply/use of plural sets of software from a number of vendors to a number of users.

As shown in FIG. 2, the distribution of object software is managed over the communications network 2 by a number of users U1, U2, ... who actually use plural sets of object software S1, S2, ... to be distributed, managed, and maintained and by a number of software vendors V1, V2, ... who supply the users with the above listed object software in the software distribution/maintenance system according to the present invention. The system comprises the following units.

(a) In each user computer 11, object software SU1, SU2, ... to be used by users U1, U2, ..., and a third process unit CP for managing plural sets of object software SU1, SU2, ...
(b) In each software vendor computer 13, vendor software libraries SL1, SL2, ... and a fourth process unit SP for providing services relating to the software libraries SL1, SL2, ...

(c) A network 12 for connecting each user computer 11 and vendor computer 13

Each unit has the following functions.

(d) The third process unit CP sends over the network 12 an object software distribution and/or maintenance request message according to an instruction of users U1, U2, ... or an instruction of a user-written program to the fourth process unit SP of the vendor computer 13 which stores a software library containing the above described object software, receives an answer message from the fourth process unit SP, and performs processes for distributing and/or maintaining the object software.

(e) The fourth process unit SP receives the message from any third process unit CP, refers to software libraries SL1, SL2, ... managed by vendors V1, V2, ... according to the received message and the settings made by vendors V1, V2, ..., generates an answer message in response to the object software distribution and/or maintenance request message, and sends the answer message to the third process unit CP of the message sender.

When any object software managed by the third process unit CP in the user computer 11 abnormally terminates during its operation, the third process unit CP detects and analyzes the abnormal condition, and then automatically sends the fault report to the fourth process unit SP of vendors V1, V2, ... over the network 12. The fourth process unit SP of vendors V1, V2, ... receives the fault report and informs vendors V1, V2, ... of the fault.

When each version of the object software is provided for a number of users U1, U2, ... by vendors V1, V2, ..., a single computer user can be used Uj of a version of software and at the same time vendor Vk of a version of another software, and furthermore, a single computer can be provided with one or more third process units CP and one or more fourth process units SP.

Each user computer 11 can be designed to store user management data UD1, UD2, ... for each of users U1, U2, ... or each user group, and the data can be set for each object software. The third process unit CP refers to user management data UD1, UD2, ... to manage one or more sets of object software SU1, SU2, ...

User management data UD1, UD2, ... can be designed to comprise user identification information of each user U1, U2, ... or each user group, software management information of each object software used by each user U1, U2, ..., software configuration information, and message process method specification information.

In specifying a method of processing a message according to user management data UD1, UD2, . . . , users U1, U2, . . . can specify a process method as a procedure for each function performed by the third process unit CP.

In specifying a method of processing a message according to user management data UD1, UD2, . . . , a vendor confirmation procedure can be used so that the vendor can be confirmed in order to protect the user software.

When the fourth process unit SP of the vendor computer 13 offers services using software libraries SL1, SL2, . . . , it can refer to vendor management data VD1, VD2 stored for each software library SL1, SL2,

Vendor management data VD1, VD2 can be designed to comprise the vendor identification information, software management information, software configuration information, message process method specification information, customer information, fault history information, etc.

When a message process method is specified using vendor management data VD1, VD2, vendors V1, V2, . . . can specify a method of the process as a procedure for each function performed by the fourth process unit SP.

When a message process method is specified using vendor management data VD1, VD2, a selection of user confirmation procedure can be implemented.

Users U1, U2, . . . can simultaneously obtain and use a plurality of versions of each object software.

General Method of Processing

The distribution of object software is managed over the communications network 2 by a number of users U1, U2, . . . who actually use plural sets of object software S1, S2, . . . to be distributed, managed, and maintained and by a number of software vendors V1, V2, . . . who supply the users with the above listed object software in the software distribution/maintenance system according to the present invention. The system comprises, in each user computer 11, object software SUI, SU2, . . . to be used by users U1, U2, . . . , and a third process unit CP for managing plural sets of object software SUI, SU2, . . . ; in each software vendor computer 13, vendor software libraries SL1, SL2, . . . and a fourth process unit SP for providing services relating to the software libraries SL1, SL2, . . . ; and a network 12 for connecting each user computer 11 and vendor computer 13. This system performs the following processes.

- (a) Users U1, U2, . . . activate the third process unit CP through a command input by users U1, U2, . . . in the user computer 11 or according to the activation through a command from a user's program.
- (b) Users U1, U2, . . . send over the network 12 an object software distribution and/or maintenance request message to the fourth process unit SP of the vendor computer 13 from which the object software is provided.
- (c) The fourth process unit SP in the vendor computer 13, from which the object software is provided, receives the message from the third process unit CP.
- (d) The fourth process unit SP refers to software libraries SL1, SL2, . . . managed by vendors V1, V2, . . . according to the received message and the settings made by vendors V1, V2, . . . , generates an answer message in response to the object software distribution and/or maintenance request message, and sends the answer message to the third process unit CP of the message sender.
- (e) The third process unit CP in the computers of the message sending users U1, U2, . . . receives the answer message from the fourth process unit SP, and performs processes for distributing and/or maintaining the above described object software according to the answer message and the settings made by users U1, U2,

When any object software managed by the third process unit CP in the user computer 11 abnormally terminates during its operation, the third process unit CP detects and analyzes the abnormal condition, and then sends the fault report message to the vendor Vk of the software over the network 12. The fourth process unit SP of vendors Vk receives the fault report message over the network 12, and informs the vendor Vk of the fault.

Management Data and Processing in a User Computer

Each user computer 11 can be designed to store user management data UD1, UD2, . . . for each of users U1, U2, . . . or each user group, and the data can be set for each object software. The third process unit CP refers to user management data UD1, UD2, . . . to manage one or more sets of object software SU1, SU2,

User management data UDj can be designed to comprise at least user identification information of user Uj or each user group, software management information of each object software S1, S2, . . . in object software group SUj used by each user Uj, software configuration information, and message process method specification information. The third process unit CP refers to user management data UDj.

In referring to a method of processing a message according to user management data UDj, user Uj can specify and use a process method as a procedure for each function performed by the third process unit CP.

In referring to the method of processing a message according to user management data UD1, UD2, . . . , a user-Uj-defined vendor confirmation procedure can be used so that the message received by the third process unit CP can be recognized as a correct message from the predetermined vendor Vk.

When user Uj activates the third process unit CP in the user computer 11 by inputting a command, the third process unit CP obtains a user name, a command name, and an object software name to generate an interactive screen displaying messages according to the input command, promotes the generation of a message in response to the command from user Uj by referring to user management data UDj of user Uj and object software Si or by invoking a user-defined process procedure, sends the generated message to the fourth process unit SP of the computer of vendors Vk of object software Si, and returns to a standby state after recording the transmission of the message.

The third process unit CP performs the following processes depending on the cause of the activation of the third process unit CP.

- (a) When the third process unit CP is activated in the user computer 11 by an input command, the third process unit CP obtains a user name, a command name, and an object software name to generate an interactive screen displaying messages according to the input command, promotes the generation of a message in response to the command from user Uj by referring to user management data UDj of user Uj and object software Si or by invoking a user-defined process procedure, sends the generated message to the fourth process unit SP of the computer of vendors Vk of object software Si, and returns to a standby state after recording the transmission of the message.
- (b) If the third process unit CP is activated in the user computer 11 by a command from a program defined by user Uj or through an abnormal termination of the object software managed by the third process unit CP, the third process unit CP obtains a user name, a command name, and an object software name to, promotes in response to the activated command the generation of a message in response to the command from user Uj by referring to user management

data UDj of user Uj and object software Si or by invoking a user-defined process procedure, sends the generated message to the fourth process unit SP of the computer of vendors Vk of object software Si, and returns to a standby state after recording the transmission of the message.

(c) If the third process unit CP is activated in the user computer 11 by a message received from the network 12, then the third process unit CP obtains from the header of the received message a destination user name, a command name, and an object software name, uses user management data UDj for the specified user Uj and object software Si, then obtains a sender name from the header of the received message, confirms that the sender is a right vendor Vk of object software Si, also confirms that the third process unit CP is in an answer-wait state, refers to user management data UDj or invokes the user-defined process procedure depending on the command in the received message, analyzes or stores the message data or stores or installs the software specified by the message, returns, in response to a specific message which requires acknowledgement, the acknowledgement or the process in response to the reception to the fourth process unit SP over the network 12, and then returns to a standby state after recording the transmission of the message.

Each time the third process unit CP is activated, a new process is generated to allow plural processes to be performed in parallel on the processes other than the process relating to a specific object software of a specific user.

Management Data and Processing in a Vendor Computer

When the fourth process unit SP in the vendor computer 13 offers services using software libraries SL1, SL2, . . . , vendor management data VDI, which is stored in each software library SLi, is referred to.

When the fourth process unit SP refers to vendor management data VDI, it refers corresponding to software library SLi to at least the vendor identification information, software management information, software configuration information, message process method specification information, customer information, update history information, and fault history information.

When a message process method is referred to using vendor management data VDI, vendor Vk can invoke a method of the process as a procedure for each function performed by the fourth process unit SP.

When a message process method is referred to using vendor management data VDI, a user confirmation procedure can be invoked to confirm the identity of user Uj who is a sender of the message, determine the qualification of the user, and return appropriate answer to the user.

In the user confirmation procedure as a part of vendor management data VDI, the qualification of a user can be determined based on commercial relations such as contracts relating to the object software, payments relating to the object software, payments relating to new services for the object software, etc.

In the user confirmation procedure as a part of vendor management data VDI, the object software can be provided and distributed only to specified users U1, U2, . . . by referring to their identification information such as the organizations to which they belong, their titles, personal names, etc.

In the user confirmation procedure as a part of vendor management data VDI, VD2, . . . , users U1, U2, . . . can be identified by passwords.

The user computer 11 and/or the vendor computer 13 should be connected to the network 12 only when it is necessary to send or receive messages to or from each other.

For example, the network connection in the way of telephone or the network transmission in the way of mail can be used properly.

The header of a message transmitted over the network 12 specifies the destination and the source of the message. Additionally, the title column of the message specifies the name of a software distribution/maintenance system, the name of a software distribution/maintenance process type, and the name of the object software.

The fourth process unit SP in the vendor computer 13 performs the following processes.

(a) The fourth process unit SP is constantly in a standby state, and activated when it receives a message from the third process unit CP over the network 12;

(b) obtains the command name and the object software name from the header of the received message, and then assigns the processes to be performed to each software Si;

(c) obtains the name of the sending user and the user identification information from the received message compares it with vendor management data VDI to confirm user Uj and determine the user Uj's access qualification relating to the command;

(d) analyzes the received message from the qualified user Uj;

(e) in response to the received command, provides summary information of software Si, provides new services for the software, supplies the updated software and the instructions to correctly update the software, retrieves information of other messages, and generates an answer message;

(f) records the summary of the answer message to user Uj, and sends the answer message to the third process unit CP of user Uj over the network 12; and

(g) returns to the standby state.

Each time the fourth process unit SP is activated, a new process is generated to allow a plurality of processes to be concurrently performed on the processes other than the process relating to a specific object software of a specific user.

Distribution and Maintenance Processes

When users U1, U2, . . . want to obtain the summary information of the object software not used by them, the following processes are performed.

In the user computer 11;

(a) User Uj enters a summary inquiry command, activates the third process unit CP in the user computer 11, sets the name of software Si, the name of vendor Vk, the network address of the vendor, an answer information storage directory, etc., and generates a summary information inquiry message.

(b) The third process unit CP sends the generated message to the fourth process unit SP of the vendor Vk.

In the vendor computer 13;

(c) The fourth process unit SP is activated when it receives the summary information inquiry message over the network 12.

(d) The fourth process unit SP refers to vendor management data VDI about object software Si cited by the summary information inquiry message to confirm the identification of the sending user Uj and determine the access qualification of the user;

(e) invokes, in response to the inquiry of the qualified user Uj, a vendor-defined procedure to provide the summary information, and generates the summary information as a summary information message; and

(f) sends the generated message to the third process unit CP of the sending user Uj over the network 12.

When users U1, U2, . . . newly obtain an object software not used by them, the following processes are performed.

In the user computer 11;

(a) User Uj enters a new installation request command, activates the third process unit CP in the user computer 11, sets the user identification information, software information of object software, vendor identification information, and installation process information, and generates a new installation request message.

(b) The third process unit CP sends the generated message to the fourth process unit SP of the vendor Vk of software Si.

In the vendor computer 13;

(c) The fourth process unit SP is activated when it receives the new installation request message over the network 12.

(d) The fourth process unit SP refers to vendor management data VDi about object software Si cited by the new installation request to confirm the identification of the sending user Uj and determine the access qualification of the user;

(e) invokes, in response to the inquiry of the qualified user Uj, a vendor-defined procedure for the new installation to determine a version Si.V.1 to be provided, obtains a set of modules of the provided version, organizes the instruction information of the installation in the user computer 11, and generates a new installation message as an answer, and (f) sends the generated new installation message as an answer to the third process unit CP of the sending user Uj over the network 12.

In the user computer 11;

(g) The third process unit CP is activated when it receives a new installation message from vendor Vk over the network 12.

(h) The third process unit CP obtains the name of a destination user and the name of an object software from the header of the received message, refers to the corresponding user management data UDj, and confirms that the third process unit CP is currently in a new installation standby state and that the sender is a correct vendor Vk;

(i) invokes a user-defined new installation process procedure, refers to the installation information in the received message, stores the software Si to be newly installed in the user computer 11, and actually installs the newly supplied software if the new installation is acceptable; and

(j) records the received message and the process result, and returns to a standby state.

When users U1, U2, . . . add or switch to a new service of an object software already used by them, the following processes are performed.

In the user computer 11;

(a) User Uj enters a new service request command, activates the third process unit CP in the user computer 11, sets the user identification information, software information of object software, vendor identification information, and installation process information, and generates a new service request message.

(b) The third process unit CP sends the generated message to the fourth process unit SP of the vendor vk.

In the vendor computer 13;

(c) The fourth process unit SP is activated when it receives the new service request message over the network 12.

(d) The fourth process unit SP refers to vendor management data VDi about object software Si cited by the new service request to confirm the identification of the sending user Uj and determine the access qualification of the user;

(e) invokes, in response to the inquiry of the qualified user Uj, a vendor-defined procedure for the supply of the new service to determine a version Si.V.1 to be provided, obtains a set of modules of the provided version, organizes the

instruction information of the installation in the user computer 11, specifies a process performed on the old service, and generates a new installation message as an answer, and (f) sends the generated new service supply message as an answer to the third process unit CP of the sending user Uj over the network 12.

In the user computer 11;

(g) The third process unit CP is activated when it receives a new service installation message from vendor Vk over the network 12.

(h) The third process unit CP obtains the name of a destination user and the name of an object software from the header of the received message, refers to the corresponding user management data UDj, and confirms that the third process unit CP is currently in a new service installation standby state and that the sender is a correct vendor Vk;

(i) invokes a user-defined process procedure, refers to the installation information in the received message, stores the software Si supplied as a new service in the user computer 11, and actually stores the new service of the software if the new service is acceptable; and

(j) records the received message and the process result, and returns to a standby state.

When users U1, U2, . . . inquire the update state of an object software already used by them of vendors V1, V2, . . . and request to send an updated software, the following processes are performed.

In the user computer 11;

(a) User Uj enters an update inquiry command, activates the third process unit CP in the user computer 11, sets the user identification information, software information of object software, vendor identification information, software management information, and post-installation process information, and generates an update inquiry message.

(b) The third process unit CP sends the generated message to the fourth process unit SP of the vendor Vk of software Si.

In the vendor computer 13;

(c) The fourth process unit SP is activated when it receives the update inquiry message over the network 12.

(d) The fourth process unit SP refers to vendor management data VDi about object software Si cited by the update inquiry to confirm the identification of the sending user Uj and determine the access qualification of the user;

(e) invokes, in response to the inquiry of the qualified user Uj, a vendor-defined update inquiry answering procedure to determine the version type Si.V to be supplied and the latest version Si.V.1, and compares it with the current version of user Uj;

(f) if the current version of user Uj matches the latest version, then generates a non-update answer message, sends it to the third process unit CP of the inquiring user Uj;

(g) if object software Si.V of the same version type as the current version of user Uj has been updated, then module delete/add instruction information and information about modules to be added are generated as an update instruction message as an answer, and the generated answer is sent to the third process unit CP of the inquiring user Uj over the network 12; and

(h) if it is determined that the current version should be switched to a newly served version of object software Si, then generates a non-update and information message about a newly served version Si.V', and sends the generated information message to the third process unit CP of the inquiring user Uj over the network 12.

In the user computer 11;

(i) The third process unit CP is activated when it receives an update instruction message or a non-update message from vendor Vk over the network 12.

(j) The third process unit CP obtains the name of a destination user and the name of an object software from the header of the received message, refers to the corresponding user management data UD_j, and confirms that the third process unit CP is currently in a standby state after inquiry and that the sender is a correct vendor Vk;

(k) if it receives a non-update message, records the received message and the process result, and returns to a standby state;

(l) if it receives an update instruction message, invokes a user-defined update reception process procedure, refers to the update instruction information in the received message, stores in the user computer 11 the software specified by the update instruction message, and actually installs the software if the specified software is acceptable; and
(m) records the received message and the process result, and returns to a standby state.

When users U1, U2, . . . inquire the update state of an object software already used by them of vendors V1, V2, . . . and request to send an updated software, the following processes are performed.

In the user computer 11;

(a) the third process unit CP in the user computer 11 is activated according to the update inquiry command instruction of a user's program, invokes an automatically-editing update inquiry procedure, refers to user management data UD_j, and generates an update inquiry message.

(b) The third process unit CP sends the generated message to the fourth process unit SP of the vendor Vk of software Si.

In the vendor computer 13;

(c) The fourth process unit SP is activated when it receives the update inquiry message over the network 12.

(d) The fourth process unit SP refers to vendor management data VDI about object software Si cited by the update inquiry to confirm the identification of the sending user U_j and determine the access qualification of the user;

(e) invokes, in response to the inquiry of the qualified user U_j, a update inquiry answering procedure to determine the version type Si.v to be supplied and its latest version Si.v1, and compares it with user U_j's current version;

(f) if the current version of user U_j matches the latest version, then generates a non-update answer message, sends it to the third process unit CP of the inquiring user U_j;

(g) if the object software of the same version type as the current version of user U_j has been updated, then module delete/add instruction information and information about modules to be added are generated as an update instruction message as an answer, and the generated answer is sent to the third process unit CP of the inquiring user U_j over the network 12; and

(h) if it is determined that the current version should be switched to a newly served version of object software Si, then generates a non-update and information message about a newly served version, and sends the generated information message to the third process unit CP of the inquiring user U_j over the network 12.

In the user computer 11;

(i) The third process unit CP is activated when it receives an update instruction message or a non-update message from vendor Vk over the network 12.

(j) The third process unit CP obtains the name of a destination user and the name of an object software from the header of the received message, refers to the corresponding user management data UD_j, and confirms that the third process unit CP is currently in a standby state after inquiry and that the sender is a correct vendor Vk;

(k) if it receives a non-update message, records the received message and the process result, and returns to a standby state;

(l) if it receives an update instruction message, invokes a user-defined update reception process procedure, refers to the update instruction information in the received message, stores in the user computer 11 the software specified by the update instruction message, and actually installs the software if the specified software is acceptable, and
(m) records the received message and the process result, and returns to a standby state.

If the third process unit CP receives a message from vendor Vk, and stores or installs in the user computer 11 new software, newly served software, or updated software received from vendor Vk, then the third process unit CP monitors the result of these processes and sends to the fourth process unit SP over the network 12 a process result confirmation message informing whether the process has terminated normally or abnormally.

Function of the Second Embodiment of the Present Invention

Function of the System

In FIG. 2, if users U1, U2, . . . activate an purchase inquiry instruction by specifying necessary software and vendors V1, V2, . . ., then the third process unit CP sends an purchase inquiry message to vendors V1, V2, . . .

The fourth process unit SP of each vendor receives the message, checks the user's qualification, automatically performs processes according to vendor management data VDI, VD2, . . ., and answers the user.

The third process unit CP which sent the purchase inquiry message receives the answer and sends it to the user. If entire software is returned, the third process unit CP stores the software according to the instruction information in the answer.

The third process unit CP refers to user management data UD1, UD2, . . . to compile and link programs and automatically install the software at a predetermined area. Furthermore, if users U1, U2, . . . want to inquire whether or not the current software used by the users has been updated, and, in case of being updated, want to request the updated version of software, the users activate the third process unit CP with an update inquiry instruction. The third process unit CP refers to user management data UD1, UD2, . . . to extract the configuration of software Si.v currently being used by the user, add user identification information, and sends to vendors V1, V2, . . . an automatic update inquiry message.

If the fourth process unit SP receives the message, it checks the qualification of the user according to vendor management data VDI, VD2, . . ., and returns software update information only to the qualified user. It returns "no update" information if the software has not been updated, and returns a method of updating the software at user site and the software itself required for the update if the software has been updated. If the third process unit CP receives the above described answer from the fourth process unit SP, then it first stores the updated software according to the answer message and based on user management data UD1, UD2, . . ., replaces the old version according to the update instruction information, etc. in the answer message with the newly stored version, and compiles and links programs if necessary to set the new version in an executable format.

The update inquiry can be activated alternatively by a user, by an automatic activation caused by the user's invoke of the object software, and by an automatic activation at a time specified by a demon program (for example, every dawn, every week, every month, etc.).

If the software managed by the third process unit CP abnormally terminates in the user computer 11, then the third process unit CP detects the cause and the state of the abnormal termination, and automatically sends a fault report message to vendors V1, V2, . . .

The fourth process unit SP records the message, reports it to the software developer, and returns a fault information acknowledgement message to the user.

Bugs are manually corrected by a software developer, and the correction is entered in the object software library of the vendor. Afterwards, an updated version is immediately provided for all the users.

As described above, the distribution and maintenance of object software is managed over the computer network 12 to which a number of users U1, U2, . . . who actually use plural sets of object software S1, S2, . . . and a number of software vendors V1, V2, . . . who supply the users with the above listed object software are connected. The software distribution/maintenance system comprises, in each user computer 11, object software SU1, SU2, . . . to be used by users U1, U2, . . ., and a third process unit CP for managing plural sets of object software SU1, SU2, . . .; and, in each software vendor computer 13, vendor software libraries SL1, SL2, . . . and a fourth process unit SP for providing services relating to the software libraries SL1, SL2, . . . The computer network 12 connects each user computer 11 to its vendor computer 13. As a result, a number of software vendors and users can distribute and use desired software, and various software developed and updated by software vendors can be distributed to users quickly and appropriately.

When any object software abnormally terminates during the operation, the third process unit CP detects and analyzes the abnormal condition, and then automatically sends the fault report to the fourth process unit SP of vendors V1, V2, . . . over the network 12. The fourth process unit SP of vendors V1, V2, . . . receives the fault report and informs vendors V1, V2, . . . of the fault. As a result, the software vendor immediately detects the occurrence of the fault of the software and can take an immediate action against the software fault.

A single computer can be used by users U1, U2, . . . of a version of software and at the same time by vendors V1, V2, . . ., and furthermore, a single computer can be provided with one or more third process units CP and one or more fourth process units SP. As a result, a computer and also a person using a computer can have a simultaneous role of a user and a vendor. This also allows, if properly authorized, easy re-distribution of software.

Each user computer 11 can be designed to store user management data UD1, UD2, . . . for each of users U1, U2, . . . or each user group, and the data can be set for each object software. The third process unit CP refers to user management data UD1, UD2, . . . to manage one or more sets of object software SU1, SU2, . . . As a result, the third process unit CP can be unified among users, and various settings can easily be defined for respective users and for respective software.

User management data UD1, UD2, . . . can be designed to comprise user identification information of each user U1, U2, . . . or each user group, software management information of each object software used by each user U1, U2, . . ., software configuration information, and message process method specification information. As a result, the third process unit CP can be unified among users, and various settings can be easily defined for respective users and for respective software.

Users U1, U2, . . . can specify a process method as a procedure for each function performed by the third process unit CP. As a result, the third process unit CP can be unified among users, and an appropriate process method can be defined for each user and for each software.

Among message processing methods in user management data UD1, UD2, . . ., a vendor confirmation procedure can be specified. As a result, the third process unit CP can be unified among users and software, and a method of confirming the vendor can be defined for each user and for each software.

When the fourth process unit SP of the vendor computer 13 offers services using software libraries SL1, SL2, . . ., it can refer to vendor management data VD1, VD2 stored for each software library SL1, SL2, . . . As a result, the fourth process unit SP can be unified for software libraries SL1, SL2, . . ., and each vendor can easily and properly define various settings for each software library.

Vendor management data VD1, VD2 can be designed to comprise the vendor identification information, software management information, software configuration information, message process method specification information, customer information, and fault history information. As a result, the fourth process unit SP can be unified for software libraries SL1, SL2, . . ., and each vendor can easily and properly define various settings for each software library.

When a message process method is specified using vendor management data VD1, VD2, vendors V1, V2, . . . can specify a method of the process as a procedure for each function performed by the fourth process unit SP. As a result, the fourth process unit SP can be unified for vendors, and each vendor can easily and properly define various settings for each process method and for each software library.

When a message process method is specified using vendor management data VD1, VD2, a user confirmation procedure can be specified. As a result, the fourth process unit SP can be unified for vendors, and each vendor can easily and properly define various settings of a user confirmation process method for each software library.

Users U1, U2, . . . can simultaneously obtain and use a plurality of versions of each object software. As a result, a user can use the software of an old version which can be easily manipulated with good stability as well as the software of a new version having new functions, thereby improving the utility of the software.

Function of the Processing Method

The distribution and maintenance of object software is managed over the communications network 12 to which a number of users U1, U2, . . . who actually use plural sets of object software S1, S2, . . . to be distributed, managed, and maintained and a number of software vendors V1, V2, . . . who supply the users with the above listed object software are connected. The software distribution/maintenance system comprises, in each user computer 11, object software SU1, SU2, . . . to be used by users U1, U2, . . ., and a third process unit CP for managing plural sets of object software SU1, SU2, . . .; in each software vendor computer 13, vendor software libraries SL1, SL2, . . . and a fourth process unit SP for providing services relating to the software libraries SL1, SL2, . . .; and a network 12 for connecting each user computer 11 and vendor computer 13. This system performs processes (a) through (e). Thus, a number of software vendors and users can distribute, maintain and use desired software, and various software developed and updated by software vendors can be distributed to and maintained at users quickly and appropriately.

When any object software managed by the third process unit CP in the user computer 11 abnormally terminates during its operation, the third process unit CP detects and analyzes the abnormal condition, and then sends the fault report message to the vendor Vk of the software over the network 12. The fourth process unit SP of vendor Vk receives the fault report message over the network 12, and informs the vendor Vk of the fault. As a result, the software vendor immediately detects the occurrence of the fault of the software and can take an immediate action against the software fault.

Function of Management Data and Processing in a User Computer

Each user computer 11 can be designed to store user management data UD1, UD2, . . . for each of users U1, U2, . . . or each user group, and the data can be set for each object software. The third process unit CP refers to user management data UD1, UD2, . . . to manage one or more sets of object software SU1, SU2, . . . As a result, the third process unit CP can be unified among users, and various settings can be easily and properly defined for respective users.

User management data UD1, UD2, . . . can be designed to comprise at least user identification information of users U1, U2, . . . or each user group, software management information of each object software used by the user, software configuration information, and message process method specification information. The third process unit CP refers to user management data UD1, UD2, . . . As a result, the third process unit CP can be unified among users, and various settings can be easily and properly defined for respective users.

In referring to a method of processing a message according to user management data UD1, UD2, . . . , users U1, U2, . . . can specify and use a process method as a procedure for each function performed by the third process unit CP. As a result, the third process unit CP can be unified among users, and an appropriate process method can be easily and properly defined for each user.

In referring to the message processing methods in user management data UD1, UD2, . . . , a user defined vendor confirmation procedure can be used so that the message received by the third process unit CP can be recognized as a correct message from the predetermined vendor Vk. As a result, the third process unit CP can be unified among users, and a method of confirming the vendor can easily and properly be defined for each user and for each software.

When users U1, U2, . . . activate the third process unit CP in the user computer 11 by inputting a command, the third process unit CP obtains a user name, a command name, and an object software name to generate an interactive screen displaying messages according to the input command, promotes the generation of a message in response to the command from user Uj by referring to user management data UDj of user Uj and object software Si or by invoking a user-defined process procedure, sends the generated message to the fourth process unit SP of the computer of vendors Vk of object software Si, and returns to a standby state after recording the transmission of the message. As a result, users can easily generate a message in an interactive mode, thereby allowing unskilled users to easily utilize software.

If the third process unit CP is activated in the user computer 11 by an input command from a user program or through an abnormal termination of the object software managed by the third process unit CP, the third process unit CP obtains a user name, a command name, and an object software name, promotes the generation of a response mes-

sage by referring to user management data UDj and object software Si or by invoking a user-defined process procedure, sends the generated message to the fourth process unit SP of the computer of the vendor Vk of object software Si, and returns to a standby state after recording the transmission of the message. As a result, the third process unit CP can be activated when a user activates object software, when the object software abnormally terminates, or when a demon program sends a command, etc. If an update inquiry is issued whenever the object software is activated, the user can always use an updated object software. If update inquiries of object software are made through the demon program, at night, in an early morning, etc., then the inquiries can be answered without a wait at, for example, the activation of the object software. Furthermore, activating the third process unit CP at, for example, an abnormal termination allows the software vendor to immediately detect bugs of the supplied software, thereby performing a quick maintenance process.

If the third process unit CP is activated in the user computer 11 by a message received from the network 12, then the third process unit CP obtains from the header of the received message a destination user name, a command name, and an object software name, uses user management data UDj for the specified user Uj and object software Si, then obtains a sender name from the header of the received message, confirms that the sender is a right vendor Vk of object software Si, also confirms that the third process unit CP is in an answer-wait state, refers to user management data UDj or invokes the user-defined process procedure depending on the command in the received message, analyzes or stores the message data or stores or installs the software specified by the message, returns, in response to a specific message which requires acknowledgement, the acknowledgement to the fourth process unit SP over the network 12, and then returns to a standby state after recording the transmission of the message. As a result, when the third process unit CP receives a message, it confirms the vendor, analyzes the message, installs the software according to the message based on user management data UD1, UD2, . . . and a user-defined process procedure.

The third process unit CP performs the processes depending on the cause of the activation of the third process unit CP. As a result, the third process unit CP can be activated at an appropriate timing.

Each time the third process unit CP is activated, a new process is generated to allow plural processes to be performed in parallel on the processes other than the process relating to a specific object software of a specific user. As a result, a plurality of processes such as purchase requests, update inquiries, etc. from a plurality of users can be concurrently performed.

Function of Management Data and Processing in a Vendor Computer

When the fourth process unit SP in the vendor computer 13 offers services using software libraries SL1, SL2, . . . , vendor management data VDI, which is stored for each software library SLi, is reference. As a result, the fourth process unit SP can be unified for software libraries SL1, SL2, . . . , and vendors can easily and properly define various settings for each software library.

When the fourth process unit SP refers to vendor management data VDI for a software library SLi, it refers to at least the vendor identification information, software management information, software configuration information, message process method specification information, customer information, update history information, and fault history information. As a result, the fourth process unit SP

can be unified for software libraries SL1, SL2, . . . and vendors can easily and properly define various settings for each software library.

When a message process method in the vendor management data VDI is referred to, vendor Vk can invoke the process method as a procedure for each function performed by the fourth process unit SP. As a result, the fourth process unit SP can be unified among vendors, and each vendor can easily and properly define various settings for a process method.

When a message process method in the vendor management data VDI is referred to, a user confirmation procedure can be invoked to confirm the identity of user Uj who is a sender of the message, determine the qualification of the user, and return appropriate answer to the user. As a result, a method of confirming and qualifying a user can be defined for each vendor, and the present invention can be used in distributing various types of software.

In the user confirmation procedure as a part of the vendor management data VDI, the qualification of a user can be determined on the basis of commercial relations such as contracts relating to the object software, payments relating to the object software, payments relating to new services for the object software, etc. As a result, the present invention can be used in distributing various types of software, including commercial product software.

In the user confirmation procedure as a part of the vendor management data VDI, the object software can be provided and distributed only to specified users U1, U2, . . . by referring to their identification information such as the organizations to which they belong, their titles, personal names, etc. As a result, the present invention can be used in distributing various types of software, including contract-base software and intra-office software.

In the user confirmation procedure as a part of the vendor management data VDI, users U1, U2, . . . can be identified by passwords. As a result, software can be protected from being distributed to an unqualified user. This makes the present invention applicable to distribute various types of software, including product software and intra-office software.

The user computer 11 and the vendor computer 13 need to be connected to the network 12 only when it sends or receives messages to or from each other. As a result, even a computer not constantly connected to the network can supply/obtain desired software, in a way similar to the communication by mails and telephones. This feature makes the present way of software distribution to be cost-effective and widely applicable.

The header of a message transmitted over the network 12 specifies the destination and the source of the message. Additionally, the title column of the message specifies the name of a software distribution/maintenance system, the name of a software distribution/maintenance process type, and the name of the object system. As a result, the destination, sender, system name, etc. can be easily informed of.

The fourth process unit SP in the vendor computer 13 performs the processes (a)–(g). As a result, when the fourth process unit SP receives a message, it determines the access qualification of a user, analyzes the message, and generates an answer message according to the received message.

Each time the fourth process unit SP is activated, a new process is generated to allow a plurality of processes to be concurrently performed on the processes other than the process relating to the same object software and the same user. As a result, a plurality of processes can be concurrently

performed on the messages received from a number of users over a network.

Function of Distribution and Maintenance Processes

When users U1, U2, . . . obtain the summary information of the object software not yet used by them, the processes (a)–(f) are performed. As a result, summary information required by qualified users U1, U2, . . . can be provided quickly and appropriately.

When users U1, U2, . . . newly obtain an object software not yet used by them, the processes (a)–(f) are performed. As a result, a newly served object software required by qualified users U1, U2, . . . can be provided quickly and appropriately. Since the new version can also be installed, an unskilled user can easily utilize an obtained object software.

When users U1, U2, . . . add or switch to a new service of an object software already used by them, the processes (a)–(j) are performed. As a result, a newly served object software required by qualified users U1, U2, . . . can be provided quickly and appropriately. Since the new service of the supplied software can also be installed, an unskilled user can easily utilize an obtained object software.

When users U1, U2, . . . inquire the update state of an object software already used by them of vendors V1, V2, . . . and request to send an updated software, the processes (a)–(m) are performed after entering a new service request command and activating the third process unit CP in the user computer 11. As a result, qualified users U1, U2, . . . can be, at any time, quickly informed of whether or not update information exists. If the object software has been updated, then the updated object software can be obtained immediately. Furthermore, since the updated software can also be installed, an unskilled user can readily utilize the object software obtained.

When the third process unit CP is activated by update inquiry command issued from the user's program, processes (a)–(n) are performed. As a result, qualified users U1, U2, . . . can automatically update object software at night or in an early morning, for example, if update information exists. Thus, a user can constantly utilize updated object software. Furthermore, since the updated software can also be installed, an unskilled user can readily utilize the object software obtained.

If the third process unit CP receives a message from vendors V1, V2, . . . and stores or installs in the user computer 11 new software, newly served software, or updated software received from vendor Vk, then the third process unit CP monitors the result of these processes and sends to the fourth process unit SP of vendors V1, V2, . . . over the network 12 a process result confirmation message informing whether the process has terminated normally or abnormally. As a result, the vendor can recognize a process result in the user computer 11.

Explanation of the Preferred Embodiments

Explanation of the First Preferred Embodiment

(1) System Configuration

FIG. 3 shows the configuration of the system according to the first embodiment. In FIG. 3, computers 21-1 and 21-2 respectively belong to users A and B. Each of the computers 21-1 and 21-2 is equipped with a client program CP for automatically updating the software in the user computer and sending fault information.

The computer of user A stores; software Si of an old configuration, and the computer of user B stores the same software Si of the latest version. The software is managed by the client program CP. As described above, software is identified as Si.V.1 in the full notation, but in the first embodiment, the software of a certain type can simply be represented as Si.1.

A computer 22 belongs to a vendor and comprises a server program SP used to automatically update software in a user computer; and a software library SL (of one type) managed by the server program SP.

A communications network 23 connects the user computers 21-1 and 21-2 to the vendor computer 22, and the client program CP and the server program SP exchange information over the network 23. The network 23 can be either a private line or a common line, establishes connections through the client program CP or the server program SP, and only has to be used during the transmission of the information.

The server program SP in the vendor computer 22 manages (specifically manages the version number and the configuration of software) the object software library SL which is obtained through development/maintenance processes performed by a vendor. If the server program SP receives from the client program CP in the user computers 21-1 and 21-2 the information of a user's current software configuration, then it compares the information with the corresponding software library SL, and automatically returns to the client program CP the update instruction information for updating user software S1 and S2 together with updated software modules.

The software library SL in the vendor computer 22 comprises a plurality of modules Ma, . . . , Mn, stores information about bug correction, update, added functions, added modules, etc. of each module, and maintains the version numbers and configurations of each module. For the sake of simplicity of explanation, simple representation is assigned to a module using symbols and numbers.

In the software library SL shown in FIG. 3, while the conventional version comprises the modules Ma, . . . , Mi, . . . , Mm, module Mi is updated into Mi* and new module Mn is added. Thus, the latest version comprises modules Ma, . . . , Mi*, . . . , Mn, Mn.

The client program CP in the user computers 21-1 and 21-2 manages the configuration of the object software. If an object software is activated, the client program CP extracts the information of the configuration of the current software (current version information) before the execution of the software, sends the information to the server program SP through the network 23, and inquires whether or not the information refers to the latest version.

Upon receipt of an answer from the server program SP, the client program CP updates the object software in the user computer if necessary according to the instruction in the answer. Then, it activates the updated software according to the user-activated instruction.

In FIG. 3, the software Si in the computer 21-1 of user A comprises modules Ma, . . . , Mi, . . . , Mm of the old configuration, while the one in the computer 21-2 of user B comprises modules Ma, . . . , Mi*, . . . , Mm, Mn of the latest configuration. These modules are managed by the client program CP.

In FIG. 3, when user A tries to activate software Si in the user computer 21-1, the client program CP detects it and inquires the server program SP in the vendor computer 22 over the network 23 by sending the information about the current version. According to the example shown in FIG. 3, software Si of the user computer 21-1 comprises modules Ma, . . . , Mi, . . . , Mm, and the configuration information is sent to the server program SP over the network 23 for inquiry.

Upon receipt of the information, the server program SP compares the information with the configuration of the software library SL, and returns the instruction information

for updating software Si of user A together with the updated software. In this example, returned is the instruction informing that module Mi is updated into Mi* and module Mn is added and the body of the updated module Mi* and added module Mn.

The client program CP of user A automatically updates software Si into the latest version using the information. According to the example, software Si in the user computer 21-1 is updated into modules Ma, . . . , Mi*, . . . , Mm, Mn. As a result, software Si of user A is updated to the latest version, just like the software Si of the computer of user B. Then, the client program CP activates the latest software according to the instruction of user A.

(2) Automatic Update of User Software

Described below is the automatic update process performed when a user activates the object software in the user computer.

(i) Process in User Computer

FIG. 4 is a flowchart showing the process in which a program stored in a user computer is automatically managed and updated in the system shown in FIG. 3. The process of a client program is described below by referring to FIG. 4.

If a user activates the object software in step S1, the client program CP managing the object software is automatically activated in step S2.

In step S3, the client program CP extracts the version information of the current object software. For example, in the case of user A shown in FIG. 3, the configuration information of modules Ma, . . . , Mi, . . . , Mm is extracted. Then, in step S4, the client program CP sends the current version information to the server program SP over the network 23.

In step S5, the client program CP receives from the server program SP the update instruction information of the object software and the updated software.

If the software has not been updated, then the update instruction information informs of no need of update. If the information tells no need of update or deletion of modules only, then no updated software is transmitted. If the software should be updated, then updated software is received, but received is only the modules to be updated into.

In the example of user A shown in FIG. 3, the instruction to delete Mi and add Mi* and insert Mn is received as the update instruction information as described above. The software of module Mi* and Mn are received as updated software.

In step S6, necessary processes are performed according to the update information. That is, if update instruction information indicates the update of modules, then specified modules are deleted from the object software and updated modules are added to the original modules. If the update instruction information tells no need of update, then no actions are taken.

Thus, the object software in the user computer matches the software library SL of the vendor.

In the example shown in FIG. 3, the object software in the computer of user A is updated into modules Ma, . . . , Mi*, . . . , Mm, Mn as described above. If the object software in the user computer is updated as described above, it is compiled and linked before the execution, if necessary, in step S7. In step S8, the client program CP is suspended after terminating its process.

In step S9, the object software is activated to start its execution according to a user's activation instruction. At this time, according to the above described steps S2-S8, it is guaranteed that the object software is the latest version of the software provided by the vendor.

(ii) Process Performed by Vendor Computers

FIG. 5 shows the process performed by a vendor computer. The process performed by the server program SP in the vendor computer 22 is described below.

In step S11, the server program is activated in the vendor computer 22 and set in a standby state.

In step S12, the server program receives an update inquiry from a client program in any user computer over the network 23, and receives the configuration information of the object software's current version. In the example shown in FIG. 3, the information received in an inquiry from user A indicates the module configuration of Ma, . . . , Mi, . . . , Mn.

In step S13, the server program SP compares the configuration with that of the object software in the software library SL managed by the server program SP, and generates update instruction information used to update the object software of the inquiring user. If the configuration of the user's current version matches the configuration in the software library SL, then the update instruction information tells "no need of update".

In the example of user A shown in FIG. 3, the update instruction information tells "delete Mi and add Mi*" and Mn*.

In step S14, the server program SP answers the inquiring client program CP with the update instruction information and the updated software required to update the user software.

As described above, the entire updated software is not sent, but only the updated or added modules are sent to the client program. In the example of user A in FIG. 3, the above described update instruction information and the software modules Mi* and Mn are returned.

In step S15, the server program SP is returned to a standby state, and control is returned to step S12 upon receipt of an inquiry. These processes are repeatedly performed.

(3) Periodical Automatic Update and User-Defined Update of User Object Software

Some users do not like to be kept waiting in updating the software to be processed according to the above embodiment of the activation of update inquiry at the time of user's activation of the object software.

In such cases, the object software can be automatically inquired for update and then automatically updated.

To attain this, a demon program can activate for the user the client program CP in the above described embodiment.

FIG. 6 shows the activation of the client program through the above described demon program. FIG. 6 differs in steps S1 and S9 from the processes shown in FIG. 4, and remains the same in steps S2 through S8.

In FIG. 6, if the demon program is activated at predetermined intervals, e.g. at night, it executes an instruction to activate the client program on the object software in step S21. Then the client program is activated in step S2, and the above described processes are performed in steps S2-S8.

If the client program is suspended in step S8, the demon program terminates in step S22.

Instead of activating a client program with a demon program, a user can directly enter a command, to activate the client program. Thus, the object software can be automatically activated whenever necessary.

(4) Automatic Transmission of Software Fault Information

The client program CP shown in the present embodiment automatically transmits to the vendor computer 22 the object software fault/bug information detected in the user computers 21-1 and 21-2 so that the information is transmitted to the developer of the software.

That is, while the object software is executed in the user computers 21-1 and 21-2, the object software may work in

an unexpected manner and terminate abnormally. This is caused by a fault/bug in the object software or in the software used by the object software.

In this case, the client program CP collects the information of conditions of the abnormal termination, such as an instruction which directly caused the abnormal termination, the state of the abnormal termination, a sequence of higher-level instructions which invoked the instruction, etc.

The above described information is immediately transmitted to the server program in the vendor computer 22 or the mail box of the developer of the software through the network 23.

The fault/bug information is received by the server program SP in the vendor computer 22, and the developer or the maintainer of the object software refers to the information for correcting the software or for developing a new feature of the software.

If a developer of the software enters a correction in a module in the software library SL stored in the vendor computer 22, then the object software in the user computers 21-1 and 21-2 can be automatically updated when the client program CP issues an update inquiry of the object software.

The fault/bug information can be transmitted through an electronic mail over the network 23.

The user computer 21-2 of user 13 according to the embodiment shown in FIG. 3 indicates the state of automatically sending the fault information. The automatically transmitted fault information received by the server program in the vendor computer 22, and the fault is reported to the developer of the software.

(5) Other Functions

The following configuration can be designed in the system according to the first embodiment.

(i) Network Connection

In the above mentioned embodiment, it is essential that a user computer and a vendor computer are connected to each other over a network. However, the connection is not required continuously. That is, a user computer only has to be connected by a client program and a server program (or transmission mail-box programs in the network which relay an inquiry from the client program or an answer from the server program) over a network from time to time just to transmit the messages in steps S4 and S5 shown in FIG. 4.

(ii) Initial Installation

If a user wants to purchase and install software provided with the above mentioned automatic update function, the software should be obtained and installed in any of the above listed methods (a), (b), and (c) of the prior art technologies.

If a network is available, the easiest method is to obtain and install a client program for software to be obtained through method (c), for example.

Activating the update inquiry of software to be obtained as if the object software were stored already in the user computer allows the client program to issue an inquiry to the vendor computer and to receive the entire module of the latest version from the vendor computer over the network. The received module can be installed and executed in the user computer.

(iii) Distribution of Non-execution Information and Automatic Update

The software to be processed according to the present embodiment can be information of the software such as explanation of new services, user handbooks, examples, etc. as well as an executable program. The information can be processed as if it were a module.

Thus, the information of the latest version can be always provided for a user by a vendor.

(6) Practical Embodiment of the Present Invention

The system shown in FIG. 7 is designed using a UNIX network connecting a computer to work stations based on the client/server system in an X window system which is a typical multiwindow system.

The system configuration and the processes performed by the client program and the server program are the same as those shown in FIG. 3.

As for a software library SL in a vendor computer 27, a directory in the UNIX file system is used as a library of the modules of the object software. Each module is used as a UNIX source file, and a file name is a unique module identification name followed by a version number as an extension.

That is, in FIG. 7, Ma.1 and Mi.2 are generated by adding the numbers 0.1 and 0.2 to the end of the module identification names Ma and Mi respectively. The software in user computers 26-1 and 26-2 is entered in a UNIX directory.

The client program CP is stored in the user computers 26-1 and 26-2 and manages the extraction of library information, update of the software, and communications to the server program SP over a network 28.

When a user activates the object software, the client program CP generates a list of file names in a directory managed by the client program CP, and transmits it to the server program SP of the vendor computer 27. The client program CP receives an answer from the server program SP of the vendor computer 27, and deletes unnecessary files and enters received files (or modules) according to an instruction of the server program SP.

The vendor computer 27 stores the server program SP which manages a library of the object software and the versions of all modules based on the development and update of the software developer.

When module configuration information of the user computer is transmitted from the client program CP, the server program SP receives the information and determines which module is required, unnecessary, or an old version in the software configuration of the client. Then, the server program SP specifies the names of the modules to be deleted and installed to generate update information as an answer to the client program CP. The update information and the modules to be installed are provided for the client program over the network 28.

The example in FIG. 7 shows that modules Ma.1, Mi.1, and Mn.1 are transmitted as the object software of the current version from the user computer 26-1.

The server program SP in the vendor computer 27 receives the information, compares the received current version information with the files in the object software library SL to determine which module is required, unnecessary, or an old version. Then, the server program SP of the vendor computer 27 generates "delete Mi.1, add Mi.2 and Mn.1" as update instruction information, and sends the update instruction information and the updated software Mi.2 and Ma.1 for update to the user computer 26-1. When the user computer 26-1 of user A receives the information, the client program CP updates the object software based on the received information.

In the example, the client program CP is provided with the function of automatically analyzing and transmitting the fault/bug information described in (4) above.

That is, if the object software in the user computers 26-1 and 26-2 abnormally terminates during the execution, it locates the direct cause of the abnormal termination and the source of the related instruction, and lists the instruction sequence associated with the related instruction. The information is described as fault/bug information.

The fault information is transmitted to the server program SP over a network. In this example, the information is transmitted by electronic mail. In the example shown in FIG. 7, the user computer 26-2 of user B automatically transmits the fault information to the vendor computer 27.

In the present embodiment, the object software actually generated and operated is multifunction and multimedia information retrieval software which is similar to a hyper text, and is described in tcl/tk language, that is, a kind of script language.

An example of fault/bug information automatically transmitted by an electronic mail from a client program of a user computer is shown.

Date; Thu. 28 Oct 93 16:36:12 JST

From; userA@ling.flab.fujitsu.co.jp (User NameA)

To; yuji@ias.flab.fujitsu.co.jp (Yuji takada)

Subject; ISISinfo bug!

Message:

couldn't find "/bin/play" to execute

ErrorInfo:

couldn't find "/bin/play" to execute while executing

"exec /bin/play - V 20 yuji10488/yuji.au"("eval" body line 1) invoked from within

"eval exec /bin/play - V 20 yuji10488/yuji.au"invoked from within

"bw View.bw1048.media.voice invoke"

... "tk-butUp.bwView.bw10488.media.voice"(command bound to event)

Data 1 describe the date, sender, destination, and occurrence of abnormal termination; data 2 describe the instruction which caused the abnormal termination, and the cause of the abnormal termination (in this example, the message tells that "/bin/play/" cannot be found); and data 3 describe higher-level instruction sequence which invoked the instruction that caused the abnormal termination.

According to the first embodiment, the functions and descriptive language of the software is not limited to the above described example, but can be optionally selected in the range of the first embodiment.

Effects of the First Embodiment

According to the first embodiment, as described above, a user computer stores a client program for managing the configuration and the execution of the object software, and a vendor computer, which is connected to the user computer over a network, stores a software library and a server program for managing a software library and automatically updates and remotely manages the software of the user computer. Therefore, problems (1)-(10) of the prior art can be successfully solved. The effects of the first embodiments can be listed corresponding to the respective problems as follows.

The effects from the viewpoint of a vendor are that:

(1) A vendor can automatically recognize the current configurations of the software in the computers of a number of users.

This facility forms the basis of software maintenance services and allows the software maintenance services to be realized efficiently and smoothly.

(2) If the vendor detects a bug and corrects the software library of the vendor, then the user who executes the object software in the user computer can automatically execute the updated software, thereby allowing the user to be immediately given the bug correction service from the vendor.

(3) If the vendor enters new services such as extended functions, new versions, etc. in the software library, then they can be immediately provided for a number of users who actually use the software.

(4) The vendor can automatically recognize who the users actually use the software are, and automatically provide them the latest software and the latest related information.

(5) Since the system of the present invention can automatically install (re-install) the software in the user computers, the vendor does not have to visit the users to install the software, but makes all services available to the users.

On the other hand, the effects from the viewpoint of a user are that:

(6) The system according to the present invention automatically installs the software and manages updated versions of the software. Therefore, the user is not required of special skills and efforts. Even a beginner or an unskilled user can be sufficiently provided with the latest services of the software.

(7) When the software in the user computer becomes faulty, the management system automatically reports the exact information of the fault to the vendor. In response to the report, the vendor removes the fault and corrects the software library. Then, the user can immediately use and execute the corrected software.

Since all the above listed processes are performed automatically except the correction process by the vendor, users are not loaded with additional work.

(8) If the vendor enters newly added functions and versions in the software library, they are automatically available to the users in the user computers when the users first use the software immediately after the entry. Thus, the users are free of extra time or jobs.

(9) The users are not required to know the information of correcting bugs, adding functions, starting a new version, etc. When these services are issued, they are automatically available to the users, thereby preventing the users from being kept using the software as an old inconvenient version.

(10) The users are free of frequent information from the vendor or a troublesome installation process pertaining to the correction of bugs and addition of functions. The users can automatically access all new services.

Summing up the above listed effects of the present invention, the users can be automatically provided with new services of a new version of the software which replaces its old version in the user computer immediately after the vendor enters in the vendor computer the correction of bugs, addition of functions, and generation of new versions for the software.

Additionally, considering that a large number of users (as many as thousands or even millions of users) use the software at various locations in the world, the effects of automatic distribution and update of the software using the software distribution/maintenance system over a network can be greatly evaluated by the vendors and users.

Explanation of the Second Preferred Embodiment

Described below in detail is the second embodiment of the present invention.

(1) System Configuration

FIG. 8 shows the system configuration according to the embodiment of the present invention. In FIG. 8, 31, 32, and 33 are user computers.

In FIG. 8, the user computer 31 stores S1.V, S2.V; the user computer 32 stores software S2.V' for user U2 and software S1.V', S3.V for user U3; and the user computers 33 stores software S1.V', S3.V'. The user computer 31 is used by user U1; the user computer 32 is used by users U2 and U3; and the user computer 33 is used by user U4. The software is managed by the system according to the present invention.

The software managed by the present system is referred to as object software Si, and i identifies the type of object software.

A communications network 35 connects the client programs of the user computers 31, 32, and 33 to the server programs of vendor computers 36 and 37.

Two vendor computers 36 and 37 are shown in FIG. 8, and the vendor computers 36 and 37 of vendors V1 and V2 are used to provide the object software of the user computers 31, 32, and 33, and to update the object software.

FIG. 8 shows a user computer and a vendor computer separately. However, a computer can be a user computer and a vendor computer simultaneously.

For example, computer A provides a specific object software for another computer B, and updates and maintains the object software. Additionally, computer A can use the object software provided by same other computer C. FIG. 9 shows a computer 38 as an example of the above described computer.

In FIG. 8, the software distribution/maintenance system according to the second embodiment comprises the following components.

(i) Server Program SP

The server program SP is stored in the vendor computers 36 and 37 to provide necessary information and software required by user Uj at his or her inquiries and requests.

According to the present embodiment, the same server programs SP is stored in the vendor computers 36 and 37.

The server program SP generates a new process each time it is activated, and a plurality of server program processes can be performed in parallel unless the processes relate to the same object software of the same user. Therefore, the server program SP can simultaneously process messages for users unless the messages relate to the same object software of the same user.

(ii) Vendor Management Data VD (Si)

The vendor management data VD refer to data including vendor identification information, software configuration information, customer management information, etc. to be stored in each software library managed by the server program, and data including information (for example, procedures) to be used by the server program SP to regulate an answering method in an answering process.

Thus, vendor management data are provided for each software library, and the server program SP uses the vendor management data to set a software-library-dependent process as a procedure, thereby allowing the server program SP to be unified.

According to the second embodiment, the follow ing information is stored in each object software Si in the vendor computers 36 and 37 of each vendor Vk (V1 and V2 in the embodiment).

Vendor Identification Information: vendor name Vk, vendor network address, etc.

Software Management Information: software name Si, location of software library SL (Si), version type and name, etc.

Software Configuration Information: module configuration for each version, etc.

Processing Method Information: user check method, password instruction method, version classification method using user management information, configuration method of install instruction information

Customer Information: user identification information, password, contract state, supplied software version, supplied history of each customer user Uj, etc.

Update History Information: development process history information, fault report information, etc.

Fault Information: fault information history, classified fault information, fault correction history, etc.

(iii) Vendor Software Library SL (Si)

A software library is managed by a vendor and stores each software Si comprising various versions Si.V.

As described above, Si indicates the type of software in each object software in a software library. For example, Si is a different software from S2. Si.V, Si.V', and Si.V'' represent the version types of the same software. For example, Si.V is developed for UNIX; Si.V' for DOS-V; and Si.V'' for MSDOS, etc.

Although not shown in FIG. 8, each of the object software Si.V, Si.V', and Si.V'' is assigned a version number (for example, Si.V.1) indicating update information. The version number is incremented each time the software is updated, for example, Si.V.1, Si.V.2, . . .

The object software Si.V comprises a number of modules Mm.1 and the module configuration of each version is set as the software configuration information in the vendor management data VD (Si). Each time a module is updated, the version number of module Mm.1 is incremented.

For example, as shown in FIG. 8, object software S1.V comprises module Ma.1, Mi.1, Mm.1, and Mi.2, and module Mi.2 indicates the updated version of module Mi.1.

The above described vendor management data VD (Si) is provided for each software type (for example, for each of software S1, S2, . . .) as shown in FIG. 8, and the server program SP refers to the vendor management data VD (Si) for each type of software to read necessary information and perform an answering process.

(iv) Client Program CP

The client program CP is stored in each user computer to send various messages from a user to a vendor, for example, a software purchase request, an update inquiry, a fault report, etc., receive an answer from the vendor, and update user software, etc.

The second embodiment shows an example of the same client program CP provided for each of the user computers 31, 32, and 33.

The client program CP as well as the server program SP generates a new process each time it is activated, and a plurality of client program processes can be performed in parallel unless the processes relate to the same object software of the same user. Therefore, the client program CP can simultaneously process requests and inquires for users unless the messages relate to the same object software of the same user.

(v) User Management Data UD (Uj)

The user management data UD are provided for each of users U1, U2, U3, and U4 managed by the client program CP, and contain user identification information, software management/configuration information, and information (procedure) regulating an installation method, etc. The user management data UD (Uj) is used to update user software when the client program CP sends various messages such as a software purchase request, update inquiry, fault report, etc. Thus, as in the case of the server program, the user management data provided for each user allow the client program CP to be unified.

According to the second embodiment, the following information is set for each user Uj.

User Identification Information: user name Uj, user network address, password, computer model, present OS, user category, etc.

Software Listing Information: list of software version names (Si.V).

The following information is provided for each version of software.

Software Management Information: software name Si, vendor identification information, version specification, software version name Si.V, contract state, etc.

Software Configuration Information: version module configuration information, etc.

Process Method Information: sending method, receiving method, sender check method, etc.

Installation process information: procedure of installation process, etc.

(vi) Object Software Si to be Used by Each User

The object software Si is stored in the user computers 31, 32, and 33, and each user uses a specific version of each software. For example, in FIG. 6, the object software used by user U1 is object software S1.V and S2.V. The object software used by user U2 is object software S2.V' different in version type from the above listed object software S2.V. Likewise, object software S1.V'' and S3.V, and object software S1.V' and S3.V'' are listed for users U3 and U4 respectively.

Typically, each version Si.V may be installed in one directory in a user computer, and each module Mm.1 may be handled as a file.

A user can own and use a plurality of versions of each object software. Thus, the user uses the object software of an old version to be used with good stability and the object software of a new version having new functions.

(vii) Network 35

The network 35 is a common UNIX network and based on an electronic mail protocol. For example, it has the message configuration.

To: VendorK@vendor.co.jp

From: Userj@flab.fujitsu.co.jp

Subject: SDMP: CheckUpdate Software Distribution System Message:

User: (User-Name: Toru Nakagawa)

Net-Address: User@flab.fujitsu.co.jp),

40 Software: (Name: Software Distribution System,

Version-Type: Standard,

Current-Version: SOS.S1.2.2.

Current-Modules: (Ma.1, Mb.3, Mc.2, Md.1, Ref.1))

The user computers 31, 32, and 33, and the vendor computers 36 and 37 do not always have to be connected to the network. That is, either one or both of the user computer and the vendor computer have to be connected to the network when a message transmitted.

That is, a message can be transmitted with both the user computer and the vendor computer connected to the network. On the other hand, as in the case of a mail system, a user sends a message to a switching station connected to a network, and a vendor periodically checks for a message and detects the arrival of the message.

(2) Functions of Software Distribution/Maintenance System

Described below is each function of the system according to the present embodiment.

(i) Initial Purchase and Installation of Software

If a user activates an purchase inquiry instruction of the client program CP described later by specifying the network address of the vendor Vk, then the client program CP sends an purchase inquiry message to the software vendor Vk. The message is automatically assigned user identification information.

The server program SP of the vendor Vk receives the message, checks the qualification of the user, automatically processes the message according to the settings of the

vendor management data VD (Si), and returns it to the user. The answering process depends on each case.

For example, the server program SP returns only summary information, demonstration program, and a payment method, returns the software with an indication clarifying the copyright, or returns the software after confirming the user using a public password.

The client program CP receives the answer and sends it to the user. If the software itself is returned, the client program CP stores the software Si.V according to the update instruction information in the answer. Then, the client program CP refers to user management data UD (Uj), compares and links programs according to the settings in the data, and automatically installs the software at a predetermined area.

(ii) Update Inquiry and Update of Software

The client program CP inquires of the software vendor whether or not the software currently used by the user has been updated or improved. If yes, the new version is automatically purchased easily as follows.

The user Uj specifies the software name Si to be inquired for update and activates the client program CP. The client program CP refers to user management data UD (Uj) to extract the configuration of the software Si.V currently used by the user, adds user identification information to the configuration, and sends an automatic update inquiry message to the vendor.

Upon receipt of the message, the server program SP checks the qualification of the user by referring to the user identification information based on the settings in the vendor management data VD (Si), and returns software update information to the user. The following answering process is performed for a qualified user.

1. If the software has not been updated, a non-update message is returned to the user. 2. If the software has been updated, then the server program SP returns an updating method applied at a user site and the software required (minimal) for update. 3. If the software is updated or improved on a large scale, the software of a new version is recommended to the user.

These answering processes can be set in the vendor management data VD (Si) by the vendor.

On receiving the answer message from the server program SP, the client program CP performs an appropriate process according to the settings in the user management data (Uj).

For example, when an updated software is returned, then the software (and added information) Si.V is temporarily stored at an area specified by the user management data UD (Uj), then the old version Si.V is replaced with the updated software according to the update instruction information in the answer message and the instruction in the user management data UD. The programs are compiled or linked if necessary to execute the new version.

The user can set the user management data UD (Uj) so that, for example, an old version can be temporarily stored as a backup version, and can be deleted after confirming that a new version works successfully.

Provided is the function of automatically reporting from the client program CP to the server program SP as to whether or not the software is normally stored and updated by the client program CP.

An update inquiry can be explicitly activated by a user, automatically activated when the software is invoked by a user, or activated at a specified time (for example, every dawn, every week, every month, etc.) by a demon program.

The explicit activation by a user is to activate an update inquiry by entering an update inquiry command. For example, a user specifies the software to be updated in an interactive operation, edits an update inquiry message, and inquires the update of the object software. The procedure of an update inquiry sending process can be set in the above described user management data UD (Uj).

According to this method, a user can update the object software at any time.

The activation of an update inquiry at the call of the software by a user is to automatically activate an update inquiry before the execution of the above described object software when the user performs a necessary process using the object software. The procedure of an update inquiry sending process is also set in the above described user management data UD (Uj). According to this method, a user can constantly use the software of the latest version.

The system can be designed such that an update inquiry can be activated when the object software terminates abnormally due to a fault/bug, etc. Thus, even if a bug is detected in the object software, a corrected object software can be automatically provided and executed if a corrected and updated version is stored in the software library SL (Si) of the vendor computer.

The activation of an update inquiry through a demon program is to periodically activate an update inquiry as a background job at night, etc. by a user at optional intervals. According to this method, unlike in the activation at the call of the software, a user is free of being kept waiting when the user issues an update inquiry.

(iii) Automatic Report of Fault Information at the Abnormal Termination of Software

When the software Si managed by the client program CP abnormally terminates during its operation, the client program CP is activated immediately. The client program detects the cause and condition of the abnormal termination (for example, higher-level instruction sequence which invoked the abnormally-causing instruction), adds the user management data UD (Uj) (only a selected portion), and automatically sends a fault report message to the vendor Vk.

The server program SP of a vendor records the fault report message after classifying it into software Si.V, and reports fault information to the software developer. Then, it returns to the user an acknowledgement message.

Bugs can be manually corrected by a software developer. If the software developer corrects a fault/bug, enters it in the object software library of a vendor, and sets version management information in the vendor management data Vd (Si), then the updated version is provided as the latest version for a user. The settings can be made such that the latest version can be published only to specified or desired users and can be provided experimentally.

(3) Communications between Client Program and Server Program, and Message Configurations and Types

The communications between the client program CP and the server program SP can be made over a network by a method similar to electronic mail and electronic data exchange EDI.

(i) Up Message: A Message Transmitted from the Client Program CP to the Server Program SP

The configuration of an up message is as follows:
Addressee: a vendor name Vk at the network address of the vendor

Sender: user name Uj at the network address of the user
Subject: present software distribution/maintenance system name, command name, software name Si

Contents: user identification information, vendor identification information, software identification information, other necessary information depending on a command, options, etc.

(ii) Down Message: A Message Transmitted from the Server Program SP to the Client Program CP

The configuration of a down message is as follows:
Addressee: user name Uj at the network address of the user
Sender: a vendor name Vk at the network address of the vendor

Subject: present software distribution/maintenance system name, command name, software name Si

Message to be returned: an up message identification information to be returned

Contents: user identification information, vendor identification information, software identification information, other necessary information depending on a command, options, etc.

(iii) Important Message Types and Functions

Important types of request/inquiry up-messages and important down-messages answering the up-messages are shown below.

<UP MESSAGE>	<DOWN MESSAGE>
SUMMARY INQUIRY	SUMMARY INFORMATION
NEW PURCHASE REQUEST	INITIAL SUPPLY
NEW SERVICE ACQUISITION REQUEST	SUPPLY OF NEW SERVICE
UPDATE INQUIRY	UPDATE INFORMATION
	NO UPDATE
COMMENT	ANSWERING
FAULT REPORT	RECEIVING FAULT REPORT
	IMPORTANT INFORMATION TO USER
	WARNING TO USER
REGULAR INSTALLATION REPORT	
ABNORMAL INSTALLATION REPORT	
TERMINATION OF USE SOFTWARE	TERMINATION, CLEARING

The types of these messages can be specified in a command name entered in the subject column of a message.

(iv) Practical Example of a Process Command in the System According to the Second Embodiment

According to the second embodiment, a process command shown in Table 1 is used as a message described above.

TABLE 1

CLASS COMMAND NAME	FUNCTION OF COMMAND
UP MESSAGE	
Inform:	INQUIRY OF SUMMARY INFORMATION, NEW SERVICE, PURCHASE METHOD, ETC.

TABLE 1-continued

CLASS COMMAND NAME	FUNCTION OF COMMAND
WantNew:	REQUEST FOR NEW PURCHASE REQUEST FOR ACQUISITION (ADDITION) OF NEW SERVICE
WantRenew:	INQUIRY OF UPDATE AND REQUEST FOR SENDING INFORMATION OF UPDATED SOFTWARE
CheckUpdate:	AUTOMATIC REPORT OF RUNTIME FAULT INFORMATION
BugReport:	MESSAGE FROM USER SUCH AS COMMENTS, COMPLAINTS, OPINIONS, INQUIRIES, ETC.
Comment:	MESSAGE OF REGULAR SOFTWARE PURCHASE AND INSTALLATION INFORMATION OF TROUBLE AT INSTALLATION OF PURCHASED SOFTWARE
InstallAck:	INFORMATION OF TERMINATION OF USE
InstallTrouble:	
FinishUse:	
DOWN MESSAGE	
ReplyInfo:	ANSWER TO INQUIRY OF SUMMARY INFORMATION, NEW SERVICE, PURCHASE METHOD, ETC.
NewInstall:	NEW SUPPLY OF SOFTWARE SUPPLY (ADDITION) OF NEW SOFTWARE SERVICE
Renew:	NO UPDATE ANSWER
NoUpdate:	INFORMATION OF UPDATED SOFTWARE
Update:	ANSWER TO USER'S COMMENTS, COMPLAINTS, OPINIONS, AND INQUIRIES
Answer:	ACKNOWLEDGEMENT OF AUTOMATIC REPORT OF FAULT INFORMATION IMPORTANT INFORMATION TO USER
BugAck:	WARNING TO USER
Notice:	INFORMATION OF CLEARING USER OBJECT SOFTWARE PROGRAM AT TERMINATION OF USE
Warning:	
Emse:	

In the system according to the present invention, an inquiry (up) message from a user is normally answered by an answer (down) message according to a corresponding important command. Furthermore, there are an exception process answer message and an purchase acknowledgement message received from a user as an acknowledgement of software.

Table 2 shows the correspondence of important commands.

TABLE 2

INQUIRY (UP)	ANSWER (DOWN) [/-EXCEPTION]	ACKNOWLEDGE (UP) [/-EXCEPTION]
Inform	ReplyInfo	(SUMMARY INFORMATION)
WantNew	NewInstall /Notice	(NEW INSTALLATION) /Install-Trouble
WantRenew	Renew /Notice	(ADDITION) /InstallAck /Install-Trouble

As shown in Table 2, the commands are normally executed as inquiries (up messages), answers (down messages), and acknowledgements (up messages).

In Table 2, a command preceded by the character / indicates an exception process

A vendor exceptionally starts sending information to a user In this case, the information is normally limited to a short message only. Applicable messages are warning messages and notice messages. These messages are normally used as a warning answer and a notice answer as an exception process in response to an inquiry from a user.

Described below in detail are the contents of the above described up messages and down messages, and associated user management data in user computers and vendor management data in vendor computers.

(a)-(c) respectively show the user identification information, the vendor identification information, and the software identification information as the contents of a message. In these figures, 0 indicates indispensable information; Δ indicates information normally required but not indispensable; and Δ indicates information required sometimes (this is also applicable in the subsequent figures).

(a) User Identification Information Section:

0 user name (name within network) required

0 user network address required

☐ user full name

☐ location of user (organization such as company, university, etc. or intra-office section)

Δ location address of user

Δ access to user location (telephone number, facsimile number, etc.)

Δ position of user

☐ user entry name (contract number, member number, etc.)

☐ user password (password, private key, etc.)

Δ state of user contract (contract type, starting date of contract, expiry date, etc.)

Δ use type of user (beginner, common user, expert, etc.)

☐ user computer (manufacturer, type, model, etc.)

Δ user software environment (type of OS, version of OS, other software)

(b) Vendor Identification Information Section:

0 vendor name (name within network) required

0 vendor network address required

☐ vendor full name

Δ location of user (organization such as company, university, etc.)

Δ location address of vendor

Δ access to vendor (telephone number, facsimile number, etc.)

Δ section and person in charge

Δ vendor entry name (contract name, name of party etc.)

Δ vendor password (password, private key, etc.)

Δ state of contract for user (contract type, starting date of contract, expiry date, etc.)

(c) Software Identification Section:

0 software name

☐ software version type name

☐ software version number

☐ software version configuration date

Δ software application object type name

Δ software application object OS name

Δ type and conditions of software application environment

Δ distribution range of software

Δ distribution rules of software

(1)-(19) respectively show the contents of other information (as messages) required for each command.

(1) Inform (Summary Inquiry) Command

Δ message: request for and inquiry of purchase

Δ message; transmitting user information for selecting version

(2) ReplyInfo (Summary Information) Command

☐ message: summary of functions, usages, effects of software

☐ message: descriptions of applicable computer type, OS, environment, etc. for software

☐ message: specification of distribution rules and contract method for software

☐ message: request for user information required to select proper version

(3) WantNew (New Purchase Request) Command

☐ message: descriptions of completion of purchase procedure and notes for process

☐ message: transmission of user information required to select proper version

(4) NewInstall (New Supply) Command

0 detailed vendor identification information

☐ password, etc. in user identification information

0 detailed software information

☐ specification of software application condition (applicable computer type, OS, environment, etc.)

0 installation instruction information of software

0 software configuration information

0 body of software

☐ document, manual, etc. of software

☐ distribution rules of software

Δ standard user procedure (invoked by CP) generated by vendor (procedure for installing, updating, and fault-reporting software)

(5) WantRenew (New Service Request) Command

☐ message: descriptions of completion of purchase procedure and notes for process

0 specification of new service acquisition request

☐ user identification information required to select proper version

☐ application for handling current software

(6) Renew (Additional Supply) Command

Δ modified portions in detailed vendor identification information

Δ password for user identification information (only when modifications are made)

0 detailed information of newly-served software (version type, number, etc.)

☐ specification of application condition of newly-served software

0 installation instruction information of newly-served software

0 configuration information of newly-served software

0 body of newly-served software

☐ document, manual, etc. of newly-served software

Δ distribution rules of newly-served software

☐ instruction and request for handling the old version of software

☐ instruction information (procedure) for handling old version of software

(7) CheckUpdate (Update Inquiry) Command

☐ user identification information (computer type, OS, environment, etc.)

0 detailed current software identification information (version type, version number, configuration date, etc.)

(8) Update (Update) Command

- ☐ explanation of software update (purpose of update, improved functions, newly solved problems, etc.)
- ☐ specification of application conditions of updated software
- 0 detailed software identification information of updated software (version type, number, etc.)
- 0 update instruction information of updated software
- 0 configuration information of updated software
- 0 body of updated software (newly added and updated portions only)
- ☐ document, manual, etc. of updated software (updated portions)
- 0 instruction information of processing old version of software

(9) NoUpdate (No Update) Command

- Δ software identification information at the latest update (version type, number, date)

(10) InstallAck (Acknowledgement of Installation) Command

- 0 identification information of received vendor message (vendor transmission time, etc.)
- Δ installation/update instruction information of installed software
- 0 process type: only when automatic installation/update is successfully performed or temporarily stored

(11) InstallTrouble (Unsuccessful Installation) Command

- 0 identification information of received vendor message (vendor transmission time, etc.)
- Δ installation/update instruction information of installed software
- 0 information of fault causing process
- Δ information as to whether or not temporary storage is performed successfully

(12) Comment (Comment) Command

- ☐ message: comments, complaints, inquiries, etc.

(13) Answer (Answer) Command

- ☐ message: answer to comments, complaints, inquiries, etc.

(14) BugReport (Fault Report) Command

- ☐ detailed user identification information (applicable computer type, OS, environment, etc.)
- 0 software identification information (version type, number, date)
- Δ software configuration information
- 0 direct phenomenon and instruction relating to abnormal termination
- ☐ instruction strings containing the instruction which caused abnormal termination

(15) BugAck (Reception of Fault Report) Command

- Δ direct phenomenon and instruction relating to abnormal termination

(16) Notice (Notice) Command

(a) Transmitted as Answer to WantNew or WantRenew Command

- ☐ message: descriptions and notes relating to purchase procedure and qualification restrictions
- ☐ message: descriptions and notes relating to software application restrictions

Δ message: information of new services of software

(b) Transmitted as Answer to CheckUpdate Command

- ☐ message: information of new services of software

Δ message: information relating to term and renewal procedure for software

- ☐ message: information relating to counteraction against specific troubles of software

(17) Warning (Warning) Command

- ☐ message: warning against illegal qualification restrictions, illegal acquisition of software, etc.
- ☐ message: information of legal acquisition of qualification, acquisition procedure, etc.

Δ message: notes and warnings against specific (problem causing) usage

(18) FinishUse (Termination of Use) Command

- 0 information of software version to be terminated

- ☐ information of process after termination of software

(19) Erase (Clear After Termination) Command

- 0 detailed object software identification information to be terminated (version type, number)

- 0 clear instruction information of object software to be terminated

- ☐ specification of termination date of object software to be terminated

FIG. 10 shows the contents of user management data in a user computer. The user management data are provided for each user. There is a single piece of user identification information in a set of management data, but the software management information and the subsequent data are provided for each software. Although not shown in FIG. 10, the user management data contain a list of the names of object software and the version names.

(a)–(d) respectively show the contents of the user identification information, the software management information, the vendor identification information, and the software configuration information respectively in the user management data.

(a) User Identification Information

0 user name (name within network) required

0 user network address required

0 user full name

0 location of user (organization such as company, university, etc. or intra-office section)

☐ location address of user

☐ access to user location (telephone number, facsimile number, etc.)

☐ position of user

0 * user entry name (contract number, member number, etc.)

☐ * user password (password, private key, etc.)

☐ * state of user contract (contract type, starting date of contract, expiry date, etc.)

Δ * use type of user (beginner, common user, expert, etc.)

0 * user computer (manufacturer, type, model, etc.)

0 * user software environment (type of OS, version of OS, other software, etc.)

(b) Software Identification Information

0 software name required

0 software version type name

0 software version number

☐ software version configuration date

☐ software application object type name

☐ software application object OS name

☐ type and conditions of software application environment

- ☐ distribution range of software
- ☐ distribution rules of software
- (c) Vendor Identification Information
- 0 vendor name (name within network) required
- 0 vendor network address required
- ☐ vendor full name
- ☐ location of user (organization such as company, university, etc.)
- ☐ address of vendor
- ☐ access to vendor (telephone number, facsimile number, etc.)
- ☐ section and person in charge
- ☐ vendor entry name (contract name, name of party, etc.)
- ☐ vendor password (password, private key, etc.)
- ☐ state of contract for user (contract type, starting date of contract, expiry date, etc.)

(d) Software Configuration Information

- 0 path indicating directory storing object software
- ☐ path indicating directory storing source program
- ☐ module configuration in directory storing source program
- ☐ path indicating directory storing object program
- ☐ module configuration in directory storing object program
- ☐ path indicating directory storing program document
- ☐ module configuration in directory storing program document

Table 3 shows the user call procedures (user processes) which can be called by the client program CP for sending a command to a server program SP or for receiving a command from a server program SP. The user management data contain a list of the names of user call procedures and their locations which the user actually provided among the above listed procedures.

TABLE 3

Command type	Automatic Process	Interactive Process
summary inquiry command (sending)	Inform-SA	Inform-SB
new purchase request command (sending)	WantNew-SA	WantNew-SB
new service supply request (sending)	WantRenew-SA	WantRenew-SB
update inquiry command (sending)	CheckUpdate-SA	CheckUpdate-SB
comment command (sending)	Comment-SA	Comment-SB
fault report command (sending)	BugReport-SA	BugReport-SB
use termination command (sending)	FinishUse-SA	FinishUse-SB
vendor confirmation procedure summary information command (receiving)	CheckVendor-A ReplyInfo-RA	CheckVendor-B ReplyInfo-RB
new supply command (receiving)	NewInstall-RA	NewInstall-RB
additional supply command (receiving)	Renew-RA	Renew-RB
update command (receiving)	Update-RA	Update-RB
no-update command (receiving)	NoUpdate-RA	NoUpdate-RB
acknowledge command (sending)	InstallAck-SA	InstallAck-SB
installation trouble command (sending)	InstallTrouble-SA	InstallTrouble-SB

TABLE 3-continued

Command type	Automatic Process	Interactive Process
answer command (receiving)	Answer-RA	Answer-RB
fault report reception command (receiving)	BugAck-RA	BugAck-RB
notice command (receiving)	Notice-RA	Notice-RB
warning command (receiving)	Warning-RA	Warning-RB
clear after termination command (receiving)	Erase-RA	Erase-RB

FIG. 11 shows the contents of the vendor management data in a vendor computer. Like the user management data, the software management information and the subsequent data are provided for each item on the list of the object software to be provided.

(a)–(d) respectively show the vendor identification information, the software management information, the software configuration information, and the user identification information data base, in the vendor management data. The user identification information data base refers to the information about all users provided with the object software.

(a) Vendor Identification Information

- 0 vendor name (name within network) required
- 0 vendor network address required
- ☐ vendor full name
- ☐ location of vendor (organization such as company, university, etc.)
- ☐ address of vendor
- ☐ * access to vendor (telephone number, facsimile number, etc.)
- ☐ * section and person in charge
- ☐ * vendor entry name (contract name, name of party etc.)
- ☐ * vendor password (password, private key, etc.)
- ☐ * states of contract for user (contract type, starting date of contract, expiry date, etc.)

NOTE: * indicates that different information can be used about the same vendor for each object software. Thus, such information is stored for each object software.

(b) Software Management Information

- 0 software name required
- 0 software version type name list
- 0 software version number list (for each version type)
- 0 software version configuration date (for each version number)
- 0 software application object type name
- 0 software application object OS name
- 0 type and conditions of software application environment
- 0 distribution range of software (conditions and procedures for qualified user)
- ☐ distribution rules of software (conditions of users)

NOTE: Version numbers are systematically assigned for respective software version types. The version provided for each user can be identified by the version number.

(c) Software Configuration Information

- 0 path indicating location of object software library to be provided
- 0 path indicating directory storing source program
- 0 list of module configuration of source program of each version

- 0 path indicating directory storing object program
 0 list of module configuration of object program of each version
 0 path indicating directory storing program document
 0 module configuration of program document of each version
 (d) Client Information in User Identification Information Database
 0 user name (name within network) required
 0 user network address required
 0 user full name
 0 location of user (organization such as company, university, etc. or intra-office section)
☐ location address of user
☐ access to user location (telephone number, facsimile number, etc.)
☐ position of user
 0 user entry name (contract number, member number, etc.)
 0 user password (password, private key, etc.)
 0 state of user contract (contract type, starting date of contract, expiry date, etc.)
 A use type of user (beginner, common user, expert, etc.)
☐ user computer (manufacturer, type, model, etc.)
☐ user software environment (type of OS, version of OS, other software)
 0 latest version type, number, and date provided for user
 A supply history to user (access date, command type, provided version, reception result at destination, record of fault report, etc.)

Table 4 shows the vendor call procedures (vendor processes) which can be called by the server program SP for receiving a command from a client program and for answering a command to the client program. The vendor management data contain a list of the names and their locations of vendor call procedures which the vendor actually provided among the above listed procedures. Vendors may have to perform different processes in a procedure for the same object software depending on the version type or version number. These differences can be properly processed by a proper logic in each procedure by performing individual processes in respective cases.

TABLE 4

Command Type	Automatic Process	Interactive Process
customer qualification confirmation procedure	CheckUser-A	Check User-B
unqualified user process procedure	NonauthUser-A	NonauthUser-B
summary inquiry command (receiving)	Inform-RA	Inform-RB
summary information command (answering)	ReplyInfo-SA	ReplyInfo-SB
new purchase request command (receiving)	WantNew-RA	WantNew-RB
new supply command (answering)	NewInstall-SA	NewInstall-SB
new service supply request (receiving)	WantRenew-RA	WantRenew-RB
additional supply command (answering)	Renew-SA	Renew-SB
update inquiry command (receiving)	CheckUpdate-RA	CheckUpdate-RB

TABLE 4-continued

Command Type	Automatic Process	Interactive Process
update command (answering)	Update-SA	Update-SB
no-update command (answering)	NoUpdate-SA	NoUpdate-SB
notice command (answering)	Notice-SA	Notice-SB
warning command (answering)	Warning-SA	Warning-SB
acknowledge command (receiving)	InstallAck-RA	InstallAck-RB
installation trouble command (receiving)	InstallTrouble-RA	InstallTrouble-RB
comment command (receiving)	Comment-RA	Comment-RB
answer command (answering)	Answer-SA	Answer-SB
fault report command (receiving)	BugReport-RA	BugReport-RB
fault report reception (answering)	BugAck-SA	BugAck-SB
use termination command (receiving)	FinishUse-RA	FinishUse-RB
clear after termination command (answering)	Erase-SA	Erase-SB

Described briefly below are the contents of the vendor software library shown in FIG. 11. The software library is used by a vendor in the software development and maintenance, and normally has a sophisticated configuration.

(a) shows an object software library; and (b) shows the contents of the object software development history/fault history data base provided for each object software library.

(a) Object Software Library

0 source program library of object software (including various version types and version numbers)

0 object program library of object software (including various version types and version numbers)

0 document library of object software (including various version types and version numbers)

(b) Contents of Object Software Development History/Fault History Database

0 development history information of object software (development date, version configuration, developed items, etc.)

0 fault history summary database of object software (initial report date, versions and modules in problem, fault phenomenon, cause of fault, correction date, corrected item, etc.)

☐ fault report database of object software (record of date, user name, version, fault phenomenon, fault causing instruction sequence of fault report from user)

(4) Inquiry Message Sending Process in a User Computer
 Described below is an embodiment of an inquiry message sending process in a user computer.

(i) Entire Configuration of a Client Program Process in a User Computer

The client program CP is stored in a user computer and manages the distribution and update of plural sets of object software used by one or more users. In response to a user's new purchase request, update inquiry, etc., the client program CP edits an up message, sends it through a network, receives an answer from the server program SP, and performs an appropriate process such as an installing operation.

FIGS. 12 and 13 shows the entire configuration of the above described processes. The process of a client program is described by referring to FIGS. 12 and 13.

In FIG. 12, the processes according to a client program are shown on the left, while the user process procedure is shown on the right. The user process procedure is set in the user management data UD (Uj) set for each user. Storing the user process procedure in the user management data UD (Uj) and invoking the user process procedure in a client program unify the client program CP of each user.

In FIG. 12, the client program CP is initialized in a user computer and set in a ready standby state (step S31 shown in FIG. 12).

The client program is activated when it receives an input command from a user or user software, or a message from the server program SP over a network (step S32).

When the client program CP is activated, then the client program CP analyzes data in a command column (or a subject column) to obtain a user name Uj, a command name, and an object software name Si (step S33). The succeeding processes, specifically the user management data (Uj) and the user process procedure UP, are performed for each user and object software.

Then, the activator of the present client program CP is identified (step S34). If it is activated by a user, control is passed to step S35. If it is activated by user software, control is passed to step S36. If it is activated over a network, control is passed to step S39.

If the client program CP is activated directly by a user, then a message edit screen is displayed in response to an input command and the user can edit the contents of the message in an interactive operation by invoking a sending procedure P1 (step S35).

In each sending procedure, retrieved is information required for each message such as a summary inquiry, new purchase request, new service supply request, update inquiry, fault report, etc. to generate a message using the information of the user management data UD (Uj).

If the user software is automatically activated, then a message is automatically edited by invoking sending procedure P1 in response to an input command (step S36).

User software is automatically activated through the above described demon program, through an instruction to start execution of object software, through an abnormal termination of object software, etc.

If an inquiry message is generated in the processes in step S35 and S36, then the client program CP sends a message to the server program SP in a vendor computer over a network (step S37).

Then, the client program CP stores the transmission of the message and returns to the standby state (step S38).

If a client program is activated after receiving a message from the network, then control is passed to step S39 shown in FIG. 13, the client program refers to the message transmission record (recorded in step S38) relating the user's software, confirms the standby state, invokes the vendor confirmation procedure P2, and confirms the message sender (in this case, the vendor).

The vendor confirmation procedure P2 is one of the user process procedures and confirms a vendor using a password, etc. from the vendor for the security of the user software.

Then, in response to the command of the received message, a user process procedure P3 of each answering process is invoked (step S40).

In each answering procedure, the software transmitted with an answer message is newly installed and updated by referring to the install instruction information in the answer message and the information in the user management data UD (Uj). If the answer message is a warning or a notice, the message is stored and displayed.

When the received message is processed, a process confirmation message is generated according to the result of the processes in the answering procedure P3 if the process result is to be automatically reported, for example, in the cases when a new supply command, an additional supply command and, or an update command is received. The generated message is sent to the server program SP of the vendor (step S41).

Then, the client program CP stores the received answer message, its process result, and a process result confirming message, and then the client program CP returns to a standby state in step S31 (step S42).

(ii) New Purchase Request Message Sending Process

Four inquiry messages actively sent by a user are Inform (summary inquiry), WantNew (new purchase request), WantRenew (new service acquisition request), and Comment (comment).

The processes of the client program CP performed when these messages are sent are performed through steps 31–35, the sending procedure P1, and step S37. Described below by referring to FIG. 12 is the process of a new purchase request, (WantNew).

When a user enters a WantNew command with a new software name Si, then a client program is activated (step S32).

The client program CP reserves an area for the software name Si in the user management data UD (Uj) of the user Uj (if a summary inquiry has already been made, the area has been reserved already) so that the subsequent processes are performed using the user management data (step S33).

The client program CP displays a message edit screen for a new purchase request, invokes the new purchase request sending procedure P1, and helps the user entering and editing necessary message portions (step S35).

Set in the new purchase request sending procedure P1 are the user identification information (user name, network address, password, etc.), software information (specification of software name, version type, etc.), vendor identification information (vendor name, vendor network address, etc.), and post-purchase process information (directory name to be stored, installing method, etc.). If the information has already been stored in the user management data UD (Uj) when the summary inquiry is made, then no re-entry is required. New information is stored in the user management data UD (Uj).

The client program CP sends to the vendor through the network a new purchase request message (WantNew message) generated by the new purchase request sending procedure P1 (step S37).

Then, the client program CP records the transmission of the message, and returns to a standby state (step S38).

(iii) Update Inquiry Message Sending Process

The CheckUpdate message for an update inquiry is sent in steps 31–37 shown in FIG. 12 by the procedure P1 in the processes shown in FIGS. 12 and 13.

The update inquiry message sending method is explained by referring to FIG. 12.

If a CheckUpdate command is entered with an update object software name specified, the client program CP activates the preparation for the update inquiry (step S32).

The CheckUpdate command can be actively entered by user as described above, and can be automatically entered at the activation of the object software and at a timing predetermined by a demon program (for example, every day, every week, every month, etc.).

The client program CP performs the following processes using the user name Uj and software name Si as keys in appropriate user management data UD (Uj) (step S33).

The activator of the present client program CP is identified in step S34. If it is activated by a user, control is passed to step S35. If it is activated by a user software, then control is passed to step S36.

If the client program CP is activated directly by a user, an update inquiry message edit screen is displayed and an update inquiry sending procedure P1 is invoked to retrieve and display information required to edit the message. At this time, the user can amend a part of the information, for example, the amendment of a version type, amendment of installation information, etc. (step S35).

The update inquiry sending procedure refers to the user management data UD (Uj), extracts the user identification information (user name, network address, password, etc.), vendor identification information (vendor name, vendor network address, etc.), software management information (version type, current version name, current module configuration, etc.), and installation information (stored directory name, backup instruction, etc.) to prepare for the generation of a CheckUpdate message.

If the CheckUpdate command is automatically activated through software, then the client program CP invokes the update inquiry sending procedure P1 and automatically sets a CheckUpdate message (step S36).

When the CheckUpdate message is generated in the processes in steps S35 and S36, the client program CP automatically sends a message to the server program SP in a vendor computer through the network (step S37).

Then, the transmission of the message is stored and the client program CP returns to a standby state (step S38).

(iv) Fault Report Message Automatic Sending Process

This process is performed in steps S31-S34, step S36, procedure P1, and steps S37 and S38 in the processes shown in FIGS. 12 and 13.

The fault report message automatic sending process is described below by referring to FIG. 12.

If the object software version Si.V managed by the client program CP abnormally terminates during its operation, a BugReport command is immediately transmitted to the client program CP (step S32).

The client program CP detects the user name and the software name which caused the abnormal termination of the software, and the following processes are performed using corresponding user management data UD (Uj) (step S33).

Since the client program CP is automatically activated, control is passed from step S34 to step S36, and the fault report sending procedure P1 is invoked to generate a fault report message.

The fault report sending procedure P1 analyzes the instruction and the state which directly caused the abnormal termination and the instruction string which invoked the instruction which directly caused the abnormal termination. The fault report sending procedure P1 extracts from the user management data UD (Uj) the user identification information, vendor information, and software identification information to generate a fault report (BugReport) message.

When a fault report (BugReport) message is generated in step S36, the client program CP automatically sends the message to the server program SP in the vendor computer through the network (step S37).

Then, the transmission of the message is stored and the client program CP returns to the standby state (step S38).

(5) Process in a Vendor Computer

Next, an embodiment of a process performed in a vendor computer is described below.

(i) Entire Configuration of a Process of a Server Program
FIGS. 14 and 15 show the entire configuration of a process of a server program. The entire configuration of a process performed in a vendor computer is explained by referring to FIGS. 14 and 15 as follows.

In FIGS. 14 and 15, processes performed by the server program are listed on the left and processes performed by vendor process procedures are written on the right. As described above, the vendor process procedure is set in the vendor management data VD (Si) for each software library. Setting the vendor process procedure in the vendor management data VD (Si) and invoking it in the server program allow the server program SP to be successfully unified.

In FIGS. 14 and 15, the server program SP in the vendor computer is constantly activated and stands by to receive various inquiry messages to be transmitted from the client programs CP of a number of users through the network (step S51 in FIG. 14).

If the server program SP receives a message from a client program CP over the network (step S52), then the server program SP checks the data in the subject column of the message to read a command name and a software name. The subsequent processes are performed for the object software thus specified (step S53).

The user name and user information are extracted from the header of the message, and the vendor management data VD (Si) of the object software are referred to in order to invoke the customer confirmation procedure specified therein (step S54).

A customer qualification confirmation procedure P6 compares the inquiring user information with the user information of the vendor management data VD (Si) and determines whether the inquiring user is a registered customer or a new customer. If the inquiring user is a registered customer, then the customer information is accessed to check the qualification of the inquiring customer.

If the inquiring user is a new customer, the user is entered as a customer and assigned, for example, a summary information access qualification, as an initial qualification for a new customer. Then, it is determined whether or not the customer is qualified for a command in the received message (step S55).

If the inquiring user is an unqualified user, then an unqualified user process procedure P7 is invoked and the specification in the vendor management data VD (Si) is referred to in order to generate an appropriate answer message indicating rejection, warning, information how to acquire qualifications, etc.

If the inquiring user is a qualified customer, the received message is analyzed and the inquiry is read (step S56).

Then, an answer procedure P8 depending on the command in the received message is invoked according to the specification of the vendor management data VD (Si) (step S57).

In response to the received command, the answer procedure P8 retrieves necessary information to return a proper answer, and obtains information required to generate an answer message. These processes include returning summary information, newly providing software, supplying new services, returning updated software information, generating other messages.

Then, an answer message is generated using above described information, and information items sent to users are stored in the user history, etc. in the vendor management data VD (Si) (step S58).

The answer message generated in step S58 is returned to the inquiry user through the network (step S59). Then, the server program returns to a standby state (step S60).

(ii) Answering Process in Response to New Purchase Request

The processes of the server program SP in the answering process in response to a new purchase request normally conform to the processes shown in FIGS. 14 and 15. In these processes, the features of a new purchase request are explained as follows.

In the customer qualification confirmation procedure P6 shown in FIG. 14, the inquiring user is determined to be a qualified user only when he or she is a user who follows a regular purchase procedure.

In the unqualified user process procedure P7, a method of following a regular procedure to make a new purchase is generated as a notice message to be returned.

In the answer procedure P8, the settings in the vendor management data VD (Si) are referred to using the user identification information in the inquiry message to determine the version type Si.V to be supplied. Then, obtained is a set of module Mm.1 forming the latest version Si.V.1 in the version type, and generated is the instruction information indicating how to perform an installation process in a user computer.

In step S58, a NewInstall message is generated as an answer message using the information in the answer procedure P8.

(iii) Answering Process in Response to Update Inquiry

Normally, this process conforms to the processes shown in FIGS. 14 and 15. In this process, the features of an update inquiry are described as follows.

In the customer qualification confirmation procedure P6 shown in FIG. 14, the inquiring user is determined to be a qualified user only when he or she is legally provided with the software and the condition of using the software (for example, the term) is still valid.

The following processes are performed in the unqualified user process procedure P7.

(a) When an inquirer is not provided with the software itself;

the procedure of newly purchasing the software is informed of with a Notice message.

(b) When the inquirer was legally provided with the software, but is not qualified after the expiration;

the procedure for the renewal of the qualification is informed of with a Notice message.

(c) When the inquirer is not legally provided with the software, that is, using it illegally;

a Warning message is returned.

In the answer procedure P8 followed in response to an update inquiry, determined are the version type Si.V' and its latest version number Si.V'.1" by comparing the user identification information in the request message with the settings in the vendor management data VD (Si).

(a) If the determined version type and number match the user's current version Si.V.1, then a NoUpdate message is returned informing that no update is required.

(b) If the user's current version is to be updated into the latest version number, then a module delete/add instruction information is transmitted to instruct that the user's current version Si.V.1 should be replaced with the latest version Si.V.1', and the required modules Mm.1' are retrieved and sent to the user.

(c) If the vendor determines that the user's current version type Si.V should be replaced with the version type Si.V' including a new service, then the information of the new version is given to the user in a Notice message.

In step S58, an answer message is generated and the information about the version which is being sent to the user

is stored in the user individual information in the vendor management data VD (Si).

(iv) Fault Report Message Receiving Process

Normally, the process conforms to the processes shown in FIGS. 14 and 15. In this process, the features of the fault report message receiving process are described as follows.

In the answer procedure P8 followed in response to a fault report, the contents of the fault report are properly classified, analyzed, stored in a fault report data base based on the vendor management data VD (Si), and then are transmitted to the developer of the software as a summary. The fault reporting user receives a BugAck message acknowledging the fault report.

In FIGS. 14 and 15, the command analysis in step S54 is always followed by the generation of a new process, and the processes performed before the transmission of an answer message in step S59 are executed by the new process. Thus, like the client program, the server program concurrently performs a plurality of processes.

(6) Answer Message Process in a User Computer

FIGS. 12 and 13 show the entire configuration of the process of the client program CP in the user computer.

If the client program CP sends various inquiry messages (refer to (4)(ii)-(iv)), the server program SP performs an answering process (refer to (5)(i)-(iv)) and returns an answer message. The client program CP receives the answer messages in the processes in steps S31-S34 shown in FIG. 12, in steps S39-S41 in FIG. 13, and in the procedures P2 and P3, and installs the purchased software, updates the user object software, updates the user object software, displays and stores Warning/Notice messages, etc.

(i) Newly Purchased Software Installation Process

Described below are the processes to be performed to receive and install a newly provided software according to the processes shown in FIGS. 12 and 13. In step S32 shown in FIG. 12, the client program CP receives a NewInstall message from the server program SP through the network.

The client program CP reads a user name Uj from the destination and a software name in the Subject column in a message, and then performs the following processes using the corresponding user management data UD (Uj) (step S33).

Since the message is received over the network in this case, control is passed to step S39 (step S34).

In step S39, the message transmission record relating to the user and the software is checked, and it is checked that a standby state is entered after the transmission of a new purchase request (WantNew) message. Then, the vendor confirmation procedure P2 is invoked to confirm that the message has been received from the right vendor. That is, in the vendor confirmation procedure P2, it is confirmed that the answer message is sent by a legal vendor V_k recorded in the user management data UD (Uj) using a password, etc.

Then, the answering procedure P3 is invoked for a NewInstall command by referring to the user management data UD (Uj) (step S40).

The procedure of installing a newly purchased software in the answering procedure P3 installs the software in the user computer using the specification in the user management data UD (Uj) and the install instruction information from the vendor. If the message is determined to be rejected, then the contents of the answer message is stored as data only, and the software is not installed. The process result is monitored to check whether it terminates normally or abnormally.

In step S41, the client program CP immediately sends the confirmation message of an installation process to the server program SP. If the installation process is normally

performed, or at least if the software is stored, the client program CP sends an InstallAck message. If the installation process abnormally terminates, the client program CP sends an InstallTrouble message.

In step S42, the client program CP records the received answer message and the transmitted confirmation message, and then returns to the standby state S31.

If a Notice message, instead of a NewInstall message is returned after issuing a new purchase request, then the client program CP performs the process described later in (iii).

(ii) Update Software Installation Process

If update software information is received and the software is installed, then the above described process (i) is performed except the following operations.

In step S32 in FIG. 12, the client program CP receives an Update message over the network.

In step S39, the message transmission record is checked to confirm that the system is in a standby state after the transmission of the update inquiry (CheckUpdate) message and the message is received from the right vendor.

The answering procedure P3 invokes an updating procedure.

The updating procedure deletes unnecessary modules in the user computer using the specification in the user management data UD (Uj) and the update instruction information from the vendor, and then stores added modules according to the received message. According to the specification of the user management data, the unnecessary modules can be actually kept undeleted, but be stored as backup data to enable both new version and old version to be used in parallel.

The software is installed in preparation of actual execution. If it is determined that the installation should not be done automatically, the software is not installed. The result of the updated version replacement process and the installation process is monitored as to whether the processes terminated normally or abnormally.

In step S41, a result confirmation message of the updated version replacement process and the installation process is immediately sent to the server program SP. If the processes terminate normally, or at least if the software is successfully stored, then an InstallAck message is returned. An Install-Trouble message is returned if the processes abnormally terminates.

If a NoUpdate message is returned in response to an update inquiry message, then no processes are performed in step S40, procedure P3, and Step S41, but answer information only is stored in step S42.

(iii) Notice Message Process

A Notice message (as information only) from a vendor is returned as an exception process in response to various inquiries from a user (new purchase request, new service acquisition request, update inquiry, comment, fault report, etc.). In this case, the client program CP responds as follows.

In step S32, the client program CP receives a Notice command through the network.

In step S33, a user name, software name, and inquiry message are read from the message destination, Subject column, and In-Reply-to column.

Control is passed from step S34 to step S39 to refer to the record of the transmission message and confirm that the answer message is received from the right vendor.

In step S40, the type of the inquiry message is explicitly indicated and the answering procedure P3 for the Notice message is invoked.

The answering procedure P3 displays the contents of the Notice message to the user, copies them to the user's mail box or stores them in the mail box.

The contents of the Notice message is interpreted to take an appropriate action in response to the standby inquiry message. For example, an answer-wait state is released for a WantNew message and a WantRenew message.

No confirmation messages are required in steps S41 and S42. The answer is recorded and the system enters the standby state.

FIG. 16 is a simplified flowchart showing the entire process of the software distribution/maintenance method according to the second embodiment. Specifically, FIG. 16 is a flowchart showing the process of the software distribution/maintenance method used in the software distribution/maintenance system according to the second embodiment. The operation is similar to the above described system operation, and can be briefly explained as follows. A client program in a user computer is activated by a user input command in step S61 or through a user program. Then, a software distribution/maintenance request message is generated in step S62 and sent to the server program of a vendor computer.

The server program receives the distribution/maintenance request message in step S63, and refers to the vendor software library in step S64 to generate and send to the client program an answer message in response to the distribution/maintenance request message sent from the client program.

The client program receives the answer message in step S65, distributes or maintains the software in step S66, and returns to the process in step S61 after the completion of the process in step S66.

FIG. 17 is a flowchart showing the entire process of the software distribution/maintenance method using user management data and vendor management data. As compared with the flowchart shown in FIG. 17, it is different in that a distribution/maintenance request message is generated by referring to the user management data in step S67, and that an answer message is generated by referring to the vendor management data in step S68.

(7) Operation of Software Distribution/Maintenance System

The software distribution/maintenance system according to the present embodiment can be applied to various ways of software distribution and maintenance. Described below are various ways of operation of the software distribution/maintenance system based on the present embodiment.

(i) Operation for Application Software Products

When the software distribution/maintenance system described above as an embodiment is applied to distribute and maintain commercial application software products, the following operation types (in consideration of vendor management data VD (Si) and user management data UD (Uj)) are introduced.

(a) The vendor determines the qualifications of users as follows.

Summary inquiry: All users can access summary information and purchase procedure. That is, every user can be informed of a contract procedure for purchasing and using software (including payment).

New purchase request: On completion of the purchase procedure, a user is assigned a unique and authorized password, and the software can be provided for a user having the authorized password.

New service acquisition request: New services can be provided for users who are assigned authorized passwords and completed new service acquisition procedure.

Update Inquiry: Accessible by a regular customer. Other users are given a Warning message.

(b) The vendor determines the types of versions as follows.

Dependency on user models: User identification information should contain the model and capacity of the user computer so that the user can be provided with software of a proper version type.

Dependency on software environment: Users are informed of the dependency on the type of operation system, language, etc. so that they can select proper version types.

(c) The standard settings in the installation process in a user computer are listed below.

The software is installed according to the installation instruction information of the vendor.

An old version can be used as backup and used with a new updated version in parallel.

(ii) Operation for the Operating System of a Large-scale Computer

When the above described software distribution/maintenance system is adopted to distribute and maintain the operating system of a large-scale computer, the following operations are introduced.

(a) The vendor determines the qualifications of users as follows.

Summary inquiry: All users are informed of summary information and required to follow the purchase contract procedure.

Other inquiries: Only customers who are assigned authorized passwords can be answered.

(b) A version type is determined between the vendor and the user under contract.

(c) Described below are the standard installation processes in a user computer.

The software transmitted with the vendor's message is stored in a secondary storage, and does not automatically activate an installation process. The installation process is performed by a user according to the sufficient instruction of the vendor.

(iii) Operation for Shareware

When the above described software distribution/maintenance system is adopted to distribute and maintain software as shareware, the following operations are introduced.

(a) The vendor determines the qualifications of users as follows.

Summary inquiry: All users are informed of the summary information and the purchase procedure. The summary information clearly mentions that the object software is shareware and demands that a regular user should pay the fund to support the development and improvement of the software.

New purchase request: Every user is sent the object software as shareware on the response of the new purchase request, and the above described fund is demanded of the user to pay after the user's satisfaction of the software.

New service acquisition request and update inquiry: Every entered user is informed of a new service acquisition request and an update inquiry. When a user issues the update inquiry certain times, the above described support fund is demanded of the user.

(b) Refer to (7) (i) above for a version type.

(c) Refer to (7) (i) above for the installation process in a user computer.

(iv) Operation for Freeware

When the above described software distribution/maintenance system is adopted to distribute and maintain software as shareware, the following operations are introduced.

(a) The vendor determines the qualifications of users as follows.

Summary inquiry: All users are informed of the summary information and the purchase procedure. The summary

information clearly mentions that the object software is freeware and should be provided for other users without deleting the indication of the copyright.

New purchase request: Every user is informed of a new purchase request, and the above description of the copyright handling is accompanied.

New service acquisition request and update inquiry: Every entered user is informed of a new service acquisition request and an update inquiry. The request and inquiry.

(b) Refer to (7) (i) above for a version type.

(c) Refer to (7) (i) above for the installation process in a user computer.

(v) Operation for Scientific Prototype Software

When the above described software distribution/maintenance system is adopted to distribute and maintain scientific prototype software, the following operations are introduced.

(a) The vendor determines the qualifications of users as follows.

Summary inquiry: All users are informed of the summary information and the purchase procedure. The summary information clearly mentions that the object software is scientific software and should be widely provided for other users with the copyright greatly respected.

New purchase request: Every user is informed of a new purchase request, and is informed that the copyright should be respected, its commercial use is prohibited, a trial use report should be presented, etc.

New service acquisition request and update inquiry: Every entered user is informed of a new service acquisition request and an update inquiry.

(b) Refer to (7) (i) above for a version type.

(c) Refer to (7) (i) above for the installation process in a user computer.

(vi) Operation for Intra-office Developed Software

When the software distribution/maintenance system is adopted to distribute and maintain intra-office-developed software, the following operations are introduced.

(a) The vendor determined the qualifications of users as follows.

Summary inquiry: Every entered member of a specified organization can be informed of the acquisition method. The information clearly mentions that other users than the members of the organization are not allowed to be provided with, use, or be presented the information. An unentered but authorizable user of the specified organization is informed of the entry method. Any user not authorizable or not of the specified organization is rejected with a warning message.

New acquisition request: Every entered member of a specified organization can be informed of the new acquisition request with the above described warning of no disclosure of the information to other users than internal members.

New service acquisition request and update inquiry: Every entered user can be informed of the new service acquisition request and the update inquiry.

(b) Refer to (7)(i) for the version type.

(c) Refer to (7)(i) for the settings in the installation process in a user computer.

It is an important nature of the software distribution/maintenance system according to the present invention that the different ways of operation for various types of software are compatible and simultaneously applicable in one system. Such operation methods are implemented in the vendor management data by the vendors independently so as to reflect their physics of software distribution.

Described below further in detail is the configuration and the process of a client program. Described first are the entire components in a user computer related to the client program.

The components are: units forming the execution environment of the client program such as an operation system, network access units, etc.; the body of the client program; standard (default) procedures to be invoked by the client program; the user management data; the user call procedures in the user management data; the data area managed by the client program; the object software storage area, etc.

1-15 respectively show the summary of the process performed by the client program. 1-15 are almost the same as the combination of FIGS. 12 and 13.

- 1 standby state
- 2 activating by user or object software, or by receiving message through the network
- 3 analyzing the command (user name, command name, and software name) and performing subsequent processes for the specified user and software
- 4 jumping according to the activator passing control to 5 when user activated CP passing control to 7 when user software activated CP passing control to 10 when message was received through network
- 5 editing a message in interactive mode for the specified command invoking and using 6, then passing control to 8
- 6 (each sending procedure)
 - retrieving and configuring information such summary inquiry, new purchase request, new service supply request, update inquiry, fault report, etc.
- 7 automatic editing message for each command invoking and using 6, then passing control to 8
- 8 sending the inquiry message after recording it
- 9 returning to the standby state 1
- 10 confirming answer-wait state, and calling and using sender (vendor) confirmation procedure 11
- 11 (vendor confirmation procedure) vendor confirmation process
- 12 invoking answering procedure for each received command, and invoking and using 13
- 13 (each answering procedure)
 - installing newly purchased software and updated software displaying and storing warning message and notes
- 14 generating and sending (only required) answer result confirmation message
- 15 recording answer and returning to standby state 1 software

An example of the process performed by the client program is described by referring to the summary of the process. The following explanation items 1-15 correspond to items 1-15 described above.

1 The OS, network capabilities, etc. are initialized in a user computer. Simultaneously, the present client program (CP) is initialized and set in a ready standby state. Also initialized are the data managed by the client program (CP).

2 The client program (CP) is activated in the following 3 cases.

- (a) A user enters a command of the software distribution/maintenance system.
- (b) The client program (CP) is activated by use's software or by invoking object software.
- (c) A message is received from the network to the software distribution/maintenance system.

These processes are provided for the CP through the OS capabilities or the network capabilities.

3 The summary of the activation is recognized through command analysis.

Obtained by the analysis are the user name, the command name, and the software name.

(a) If a user inputs the command, the user specifies the command name and the object software name. The OS can supplement the user name.

(b) If user's software or object software is activated, then the command name and the object software name are specified in the activation instruction. The user name is supplemented by the OS.

(c) If a message is received through the network, then the user name is obtained from the destination of the message, and the command name and the object software name are obtained from the Subject column of the message.

As a result of the analysis, the user management data (including the user-specified procedure) for the object software of the user are referred to in the subsequent processes.

4 Jumping according to the activation

In a sequential execution of the client program CP processes, a jump is made to 5 by a user activation; 7 by a user software activation; and 10 by the message from the network.

In a parallel execution of the CP processes, the result of the command analysis is recorded in the CP operation history list and a child process is generated and executed concurrently with the parent CP program. If another child process is being performed for the same user and the same object software, the new-born child process is kept waiting until the preceding child process completes.

5 When a user inputted the command;

The client program (CP) (or the child process of the client program in the parallel execution) invokes an interactive inquiry editing procedure 6 for the command. If the procedure is specified in the user management data, the user-specified procedure is used. If not, the default is the standard procedure. By use of the procedure, the user edits and generates a transmission message for the specified command in an interactive mode. Then, the control jumps to 8.

7 When the client program was activated by software or a message;

The client program (CP) (or the child process of the client program in the parallel execution) invokes an automatic inquiry editing procedure for the command. If the procedure is specified in the user management data, the user-specified procedure is used. Otherwise, the default is the standard procedure. The procedure refers to the user management data and automatically edits and generates an inquiry message for the specified command.

8 The message generated in 5 or 7 above is recorded and then sent to the vendor's computer through the network.

9 The client program (CP) returns to the standby state 1.

The client program (CP) (or the child process of the client program in the parallel execution) records the transmission on the CP process history list. In the parallel execution of processes, the child process terminates its job and is deleted. The client program (CP) returns to the standby state 1.

10 When the message was received through the network;

The client program (CP) (or the child process of the client program in the parallel execution) refers to the CP process history list and confirms that the user sent a message relating to the object software and that the received message can be expected as an answer.

To confirm that the sender of the received message is the right vendor, the client program (CP) or its child process invokes the vendor confirmation procedure 11. If the procedure is specified in the user management data, then the user-specified procedure is used. Otherwise, the default is the standard procedure.

If the received message is rejected by the confirmation, then the client program (CP) terminates without performing the subsequent processes and returns to the standby state 1.

12 Answer receiving procedure in response to a right answer

The client program (CP)(or the child process of the client program in the parallel execution) invokes answer receiving procedure 13 for the command. At this time, either interactive or automatic answer receiving procedure is selected depending on whether the corresponding inquiry message was processed interactively or automatically. If the answering procedure is specified in the user management data, then the user-specified answer receiving procedure is used. Otherwise, the default is the standard procedure.

14 If the command requests the confirmation of the result of the answer receiving process (that is, either NewInstall, Renew, or Update command), then the client program (CP)(or its child process in the parallel execution) automatically monitors the result of the answer receiving process in 13. If the answer receiving process in 13 (storing and installing the software) has been successfully performed, then an acknowledgement (InstlAck) message is generated (by the user-specified or the default automatic editing procedure) and sent to the vendor through the network. If the answer receiving process 13 abnormally terminates, an InstallTrouble message is generated (by the user-specified or the default automatic editing procedure) and sent to the vendor through the network.

15 The answering process is recorded and the program returns to the standby state 1.

The client program (CP) (or its child process in the parallel execution) records the answer receiving process (and the transmission of the acknowledgement trouble message in the case of 14 above) on the CP process history list. In the parallel execution of processes, the child process terminates its job and is deleted. The client program (CP) returns to the standby state 1.

(a)-(h) explain 6, that is, explains the processes in various inquiry editing procedures. The configurations of the messages generated by these procedures are described above. Common Process

(1) obtaining and setting user identification information by referring to user management data

(2) obtaining and setting vendor identification information by referring to user management data

(3) obtaining and setting software identification information by referring to user management data

(4) generating header of a transmission message (destination, source, and subject columns)

(b) Inform-SA/SB (sending summary inquiry) (done in the common process)

(c) WantNew-SA/SB (sending new purchase request) (done in the common process)

(d) WantRenew-SA/SB (sending new service request) specifying new service supply request

(e) CheckUpdate-SA/SB (sending update inquiry)
Detailed current software identification information (version type, version number, configuration date, etc.) is extracted and specified by referring to user management data (or the current object software if necessary).

(f) Comment-SA/SB (sending comments) writing any comment

(g) BugReport-SA/SB (sending fault report)
Software identification information (version type, number, date) is extracted, and the cause and instruction directly involved in an abnormal termination is extracted and analyzed.

(h) FinishUse-SA/SB (sending termination of use message) The version information of terminated software is extracted.

(a)-(h) explain only the main processes to be done in the procedures. In the automatic editing procedures, the minimal essence of a message is prepared, while more detailed portion can be specified in the interactive editing procedures.

(a)-(j) explain the processes performed by various answering process procedures.

a) ReplyInfo-RA/RB (Receiving Summary Information)
The contents of summary information are stored in a predetermined directory.

b) NewInstall-RA/RB (Receiving Newly Supplied Software)

Detailed vendor identification information, a password in user identification information, detailed software identification information (version type, number, date, etc.) are stored in user management data.

The following received data of software is temporarily stored in a predetermined directory:

specified software application condition (applicable computer type,

OS, environment, etc.); software installation instruction information; software configuration information; body of software; software document and manual; software distribution rules; standard user procedure (to be invoked by CP) generated by vendor.

After checking various application conditions specified by vendor, the body of software is installed according to the installation instruction information.

The results of the storage and installation as to whether the processes are successfully performed or abnormally terminate should be monitored.

(c) Renew-RA/RB (Receiving Additional Services)
The newly served software is processed as in the case of the above described NewInstall-RA/RB. The old version of the software should also be properly handled (i.e. either deleted or kept).

(d) Update-RA/RB (Receiving Update)
The following software update information is stored in a predetermined directory:

software identification information (version type, number, date);

explanation of software update (purpose of update, improved functions, newly solved problems, etc.);

specification of application conditions of updated software; detailed software identification information of updated software (version type, number, etc.);

update instruction information of updated software; configuration information of updated software; body of updated software (only updated and added portions); documents, manuals, etc. of updated software (updated portions); and instruction information for processing an old version of software.

The above listed information is checked, and the current object software is updated based on the update information according to the update instruction information specified by the vendor.

Monitored is the storage and update processes as to whether the processes are successfully performed or abnormally terminate.

(e) NoUpdate-RA/RB (Receiving No Update Message)
No action is taken.

(f) Notice-RA/RB (Receiving Notice Message)
The contents of the received message are temporarily stored in the directory and displayed.

(g) Warning-RA/RB (Receiving Warning Message)
The contents of the answer is temporarily stored in the directory and displayed immediately.

(h) Answer-RA/RB (Receiving Answer)

The contents of the answer is temporarily stored in the directory and displayed.

(i) BugAck-RA/RB (Receiving Fault Report)

The contents of the answer is temporarily stored in the directory and displayed.

(j) Erase-RA/RB (Receiving Terminate And Erase Message)

The contents of the answer is temporarily stored in the directory to be checked, and the information of the object software is cleared according to the clear instruction information.

(a) and (b) respectively explain 14, that is, explains the processes performed in the acknowledgement message sending procedure as a result of the answering process.

a) InstallAck-SA/SB (Sending Acknowledgement)

The identification information of a received vendor message (vendor transmission time, etc.) is extracted, and a message is generated according to the result of monitoring the installation and updating the software to inform that a temporary storage or process has been successfully performed.

(b) InstallTrouble-SA/SB (sending installation Trouble message)

The identification information of the received vendor message (vendor transmission time, etc.) is extracted, and a message is generated according to the result of monitoring the installation and updating the software to inform of the occurrence process of the fault and a process to be performed.

Then, the configuration and the processes of the server program are described as follows. In a vendor computer, there are following components related to the server program: the execution environment of the server program including the operation system; the body of the server program; (default) procedures to be invoked by the server program; the vendor management data; the vendor call procedure in the vendor management data; the data area managed by the server program; the object software storage area, etc.

1-14 respectively show the summary of the process performed by the server program. The summary of the process is similar to the configuration of the process for the server program explained in FIGS. 14 and 15.

1 standby state

2 receiving a message from client program over network

3 analyzing the command of the message (user name, command name, and object software name)

and performing subsequent processes for specified object software

4 interpreting and storing the received message and invoking procedure 5 to confirm user information and user qualification

5 performing customer qualification confirmation procedure as follows;

extracting inquirer user information,

comparing extracted data with vendor management data, confirming customer information for authorized customer and

entering a new customer with assigning initial qualification,

determining command access qualification of the customer

6 in case the user is invoking unqualified user process procedure 7

7 generating a message of rejection, warning, qualification obtaining method, etc. and passing control to 13

8 in case the user is qualified, invoking a receiving procedure 9 for the specified command, analyzing the received message, and determining a command to be returned

10 invoking an answering procedure for the command to be returned

11 performing the answering procedure, for the preparation for returning summary information, newly providing software, providing new service, returning update software information, returning other messages, etc.

12 configuring the answer message and recording it

13 sending the answer message

14 returning to the standby state 1.

An example of the process performed by the server program is explained below. The following items 1-14 correspond to 1-14 shown above. In this example, a process of receiving a message from the client program is separate from a process of returning an answer message in response to the received message, so as to invoke the receiving procedure 9 and the answering procedure 11 respectively. But obviously, a receiving/answering procedure can be designed to perform these processes in series.

1 The operating system and the network capabilities are initialized in a server computer. Simultaneously, the server program (SP) is initialized. After the data managed by the server program (SP) are initialized, the program is set in a ready standby state.

2 When a message to the software distribution/maintenance system is received through the network, it is transmitted to the SP through the OS capabilities or the network capabilities, thereby activating the SP.

3 The header of the message is analyzed to obtain the user name from the message source, and the command name and the object software name from the subject column. Used in the subsequent processes are the vendor management data (including the vendor-specified procedures) for the object software specified by this command analysis.

In the parallel execution of SP processes, the command analysis result is recorded on the SP process history list. Then, a new child process is generated and executed concurrently with the parent SP program. If another child process is being performed for the same object software and for the same user, the new-born child process is kept waiting until the preceding child process completes.

4 The server program (SP) (or a child process of the server program in the parallel execution of SP processes) first reads the entire received message, interprets the message, and temporarily stores it in a storage area of the server program (SP) (or the child process). The contents of the message can be user identification information, vendor identification information, software identification information, and various information required or optional for each inquiry/request command. Then, using the information, the customer qualification confirmation procedure is invoked to confirm the qualification of the user. If the procedure is specified in the vendor management data, then the vendor-specified procedure is used. Otherwise, the default is the standard procedure.

5 In the customer qualification confirmation procedure, the user identification information in the received message is compared with the user identification information data base in the vendor management data to determine the access qualification of the user. The methods of comparing and determining the related data are set by the vendors according to their distribution policy, as described later. If the user is a new customer, the range of the software to be provided for the user and the conditions of the access to

- the provided software should be clearly defined. If the user is an authorized user, his or her password and access right to each command should be confirmed.
- 6 The vendor-defined (or default) unqualified-user processing procedure 7 is invoked to handle a user who has been determined to be unqualified in the previous step. A message informing rejection, warning, instruction to acquire the qualification, etc. is returned to the unqualified user according to the vendor-defined rules. After returning the message, control is passed to 13.
 - 8 In response to the message received from the qualified user, the program analyzes the received message and interprets the contents of the inquiry from the user. To attain this, the vendor-defined (or default) receiving procedure 9 is used for each command of the received message. In the receiving procedure, the contents of the received message are interpreted and determined by referring to the vendor management data to select the type (normally a single type, but possibly plural types) of the command to be returned.
 - 10 In response to the above described determination, the vendor-defined (or default) answering procedure is invoked to answer the command.
 - 11 In the answering procedure, an answer message is generated by referring to the contents of the received message, the vendor management data, and the vendor object software library. An answer message contains, for example, in case of an Update message, the update instruction information to update the software and the updated portion of the software.
 - 12 The answer message is completed with headers etc. and recorded in the SP process history list.
 - 13 The answer message is sent to the inquiring client program through the network.
 - 14 (The child process is deleted here in the parallel execution of processes.) The server program (SP) returns to the standby state 1. The customer qualification confirmation procedure 5 and the unqualified user process procedure 7 are used to practically realize various vendor-defined software providing policy. The access qualifications of customers are regulated at four levels, that is, (0)-(3) as shown in Table 5. Access by a user to certain object software is limited to the inquiry/request commands shown in Table 5 depending on the four levels. There is a choice by the vendor, as usual for any commercial sales policy, between the pay-first supply-later or the supply-first pay-later policies in setting the customer qualification confirmation rules at level (2) for a new software request and a new service request command.

Access Qualification Level	Inquiry/Request Command Allowed
(0) no access permitted	none
(1) accessible to summary information only	summary inquiry (Inform)
(2) accessible to request for new software and new service	new purchase request (WantNew) new service request (WantRenew)
(3) access qualification of current version	update inquiry (CheckUpdate) comment (Comment) fault report (BugReport) termination of use (FinishUse)

The information required for the determination of the access qualifications includes in addition to the basis ones like user identification information and software identification information, some information depending on each command, such as information about a payment process and rules, information about version selection, etc.

The software supply rules as the contents of the customer qualification confirmation procedure 5 and the unqualified user processing procedure 7 depend on vendor's operational policy in distributing the software. Five operation methods (a)-(e) are described below as examples.

(a) Operation Method for an OS in a Large-scale Computer Services are normally provided for the limited range of authorized users.

Summary inquiries (level (1)) are answered in a wide range.

Services of levels (2) and (3) are available for a limited number of users who made individual agreements. Therefore, the vendor enters authorized users after agreements to process the users' messages based on a strict password system.

(b) Operation Method for Commercial Application Software Products

All users are positively answered when they send summary inquiries (level (1)). All accessing users are entered in the customer data base of the vendor; the summary information is arranged and returned in a format suitable for each user; and purchase and payment methods are specified.

Relating to a new purchase request at level (2), answered are those who have promised to pay for the software (for example, by specifying their bank account number credit card number in an official format) or who have completed the payment for the software. Other users are answered with a description of payment method without providing the software. New service acquisition requests are answered likewise.

For an inquiry at level (3), the user identification information such as the user name, entry number, password, computer model number, etc. is required and compared with the data in the vendor management data in order to identify the user as an authorized user who completed the payment. A user who is determined to be unqualified is so recorded and his or her request is rejected with a warning message. (The vendor can analyze the warning records later and legally prove that a specific user illegally obtains and uses the software of the vendor for a long period.)

(c) Operation Method for Shareware

All users who send inquiries at level (1) are positively answered with summary information, message which clearly mentions with the purchase method and the charge as shareware.

For a new purchase request, the requester is entered as a user, immediately provided with the software, and requested to pay the charge later.

For an inquiry at level (3), the inquirer is checked for his or her user entry (an unentered user is newly entered) and positively answered. If the vendor finds that the inquirer has not paid the charge even after certain times of update inquiries for a certain period, then the inquirer can be urged to pay the charge for the software as shareware (by use of a Notice message, etc.).

(d) Operation Method for Scientific Software

A message at level (1) is answered in a wide range (mostly for scientific users). An inquirer is answered with summary information of the scientific prototype software and its distribution policy such as the respect of copyright, prohibition of commercial use of the software, and a request for a trial use report.

For a request at level (2), the requester is entered as a user, provided with the software, and required to accept the distribution rules. Normally, the software is provided free of charge.

For an inquiry at level (3), the entry of the inquirer is checked (an unentered user is newly entered) and answered

positively. After the confirmation of a number of his or her inquiries and requests, the inquirer can be requested to present a trial use report.

(e) Operation Method for Intra-office Software

A strict check is made for an inquiry at level (1). Any access is immediately checked as to whether or not the user is a regular member of the organization, and the access may be limited to some range of members according to his or her position, department, and name. An unqualified inquirer is rejected and answered with a warning message.

For a message or inquiry at level (2) or (3), an authorized user at level (1) is positively answered in most cases. The department and name of the user are confirmed and the user is allowed to access using his or her password for security.

(a)-(f) respectively explain the receiving procedures 9 and the answering procedures 11. Each of the procedures for various types of received commands is primarily specified by the vendor. Normally, the processes are automatically performed, but difficult problems may be determined by the vendor in an interactive edit mode. Though the receiving procedures and the answering procedures are explained separately below, they can be designed to be processed in series as unified receiving/answering procedures.

(a) Common Process in Receiving Procedures

The contents of the received message have been read in step S4 in the server program (SP) and stored. Using the stored data, the basic portions of the received message, that is, user identification information, vendor identification information, and software identification information, are compared with the records in the vendor management data in order to complement the information required in the following processes and to update the record in the vendor management data.

(b) Inform-RA/RB (Receiving Summary Inquiry)

User identification information and software identification information are referred to, and the communication text in a received message, if any, is interpreted so that the answering process specifies a ReplyInfo command.

(c) ReplyInfo-SA/SB (summary information returning Process)

The following information of specified object software (for specified version type) is extracted from the vendor management data and the software library to edit in the format of communications text.

- ☐ summary of functions, usages, effects of the software
- ☐ descriptions of applicable computer type, OS, environment, etc. for the software
- ☐ specification of distribution policy and contract method for the software
- ☐ request for user information required to select proper version type of the software

(d) WantNew-RA/RB (Receiving New Purchase Request)

User information is interpreted and the vendor management data is referred to in order to determine the optimum version type for the user, and to specify a NewInstall command in the answer message. A Notice command can be returned as an exception. For example, if there are no available versions in a user environment, the Notice command is returned to explain the current state. If the user requests an old version, then the Notice command is returned to recommend a new version.

(e) NewInstall-SASB (New Supply Answer)

The latest version of the specified version type of the software is newly supplied. The information in the vendor management data and especially the information in the vendor's software library are copied to be edited as an answer message.

- ☐ detailed vendor identification information
- ☐ password, etc. in user identification information (assigned separately in a secured method)
- ☐ detailed software information
- ☐ specification of software application condition (applicable computer type, OS, environment, etc.)
- ☐ installation instruction information of the software
- ☐ software configuration information
- ☐ body of the software
- ☐ document, manual, etc. of the software
- ☐ distribution rules for the software

Δ standard user procedures (to be used by the client program) generated by vendor (procedures for installing, updating, and fault-reporting for the newly supplied software)

(f) WantRenew-RA/RB (Receiving New Service Request)

Whether or not a newly served software is provided is determined in a way similar to process of receiving a WantNew command. At this time, the treatment of the old software stored in the user computer is determined (by referring to the vendor management data). If a new service is not applicable a Notice command is returned as a response.

(g) Renew-SA/SB (Additional Supply Answer)

By extracting from the vendor management data and particularly the vendor software library, the following information is edited as an answer to provide the specified software (new service).

Δ modified items in detailed vendor identification information

Δ password for user identification information (only when modifications are made)

- ☐ detailed information of the newly-served software (version type, number, etc.)
- ☐ specification of application condition of the newly-served software
- ☐ installation instruction information of the newly-served software
- ☐ configuration information of the newly-served software
- ☐ body of the newly-served software
- ☐ document, manual, etc. of the newly-served software
- Δ distribution rules of the newly-served software
- ☐ instruction and request for treating the old version of the software
- ☐ instruction information of processing the old version of the software

(h) CheckUpdate-RA/RB (Receiving Update Inquiry)

The detailed information in the user's current software version (version type, version number, configuration data, etc.) is compared with the information in the latest version of the vendor management data, and it is determined whether or not the software needs to be updated. At this time, if the user identification information in the received message (computer type, OS, environment, etc.) is found, changed from the record in the vendor management data (changed at a user site), or if the criterion of the version selection is changed by the vendor, then an appropriate version type should be selected.

In a normal update process (including a small-scale change to another version type), an Update command is specified as the answer. If no update has been made, then, a NoUpdate command is chosen as the answer message. If a

large-scale change in version type or a change into a new service is recommended, a Notice command (not an Update command) is chosen as the answer message.

(i) Update-SA/SB (Update Information Answer)

The following information is edited as an answer to update from user's current software version to the specified version.

- ☐ explanation of updating software (purpose of update, improved functions, newly solved problems, fixed bugs, etc.)
- ☐ specification of application conditions of the updated software
- ☐ detailed software identification information of the updated software (version type, number, etc.)
- ☐ update instruction information of the updated software
- ☐ configuration information of the updated software
- ☐ body of the updated software (newly added and updated portions only)
- ☐ document, manual, etc. of the updated software (updated portions)
- ☐ instruction information of processing the old version of the software (deletion through replacement, reserving for a predetermined period, etc.)

(j) NoUpdate-SA/SB (Answering Without Update)

A NoUpdate command is specified in an answer message.

A software identification information at the latest update (version type, number, date)

(k) Notice-SA/SB (Note Answer)

The following messages are edited (for each case) according to the results of the receiving process.

descriptions and notes relating to purchase procedure and qualification restrictions

information relating to term and renewal procedure for software

descriptions and notes relating to software application restrictions

information relating to counteraction against specific troubles of the software

information of new services of the software

(l) Warning-SA/SB (Warning Answer)

A Warning message is issued from the unqualified user process procedures in most cases, but can also be issued as a warning by answering procedure as an exception in an emergency.

For example;

if a user uses the object software in an undesirable environment and a serious fault is expected unless proper correction is made.

if a renewal procedure is recommended because the expiry is closing down.

if a serious fault is detected in the vendor software and restrictions of use should be placed on the user.

An answer message is generated depending on the above listed conditions.

(m) Install-Ack-RA/RB (Receiving Acknowledgement)

The acknowledgement is recorded in the software supply history of the user in the vendor management data. No answering process is performed.

(n) Install-Trouble-RA/RB (Receiving Installation Trouble Message)

The message is recorded in the software supply history of the user in the vendor management data, and is shown to the vendor's engineer in charge.

(o) Comment-RA/RB (Receiving Comment)

The message is recorded in the software supply history of the user in the vendor management data, and displayed in an interactive mode to ask the vendor for an appropriate processing. An Answer command is used in the answering process.

(p) Answer-RA/RB (Answering Process)

In response to a user's Comment command, an answer message is generated in an interactive mode by the vendor to answer various questions and complaints.

(q) BugReport-RA/RB (Receiving Fault Report)

The contents of the received message are recorded in the fault report history database and transmitted to the vendor. Specifically, if similar fault reports are not stored in the fault report history database and the fault is determined to be a new type, then a warning message is given to the vendor's engineer in charge. Normally, a BugAck command is used in the answer message.

(r) BugAck-RA/RB (Fault Report Receipt Answer)

A regular message just indicating the receipt of a fault report is edited automatically.

(s) FinishUse-RA/RB (Receiving use Termination)

The message is recorded in the software supply history of the user in the vendor management data. Commercial and technical procedures are determined for the use termination of the specified software version. If the termination is appropriate, then a terminate-and-clear answering procedure is specified.

(t) Erase-SA/SB (Terminate-and-clear Answer)

Clear instruction information is edited by specifying the identification information for the specified software version. An expiry date of the use of the object software to be cleared may be specified at this time.

Described finally are practical examples of using the software distribution and maintenance system according to the second embodiment of the present invention. The present system aims at being used widely as a general-purpose system and allows to be implemented with sophisticated technologies at vendor computers and user computers. Explained below, however, is the second embodiment realized with rather simple well-known element technologies. It is noticed that the present invention is not limited by such element technologies.

(a) Simple Practical Embodiment of Configuration of Software and Generation of Update Information (1)

First, an object software S1 comprises one version type VTI containing a plurality of modules Ma, Mb, Mc,

(a1) Configuration and Management of Software in Vendor Computer

In a vendor computer, the software library S1 of the software (for example, in a UNIX system) is stored under a single directory named, for example, directory S1.VTI. The source program of each of modules Ma, Mb, Mc, . . . is put as a single file in the directory. Each module is represented by a module name (Ma, Mb, etc.) followed by its version number (1, 2, . . .) as a complete file name (for example, Ma.1, Mb.3, etc.). The version number of each module is incremented by 1 each time it is amended or updated.

The easiest method of managing the versions of a software library is to always store a single set of the latest modules. If a module or a set of modules is updated, the version number of the software is incremented by 1. According to this method, the software version and the internal configuration of the modules are as follows, for example.

Software Version	Module Configuration				
S1.VT1.1	Ma.1	Mb.1	Mc.1	Md.1	
S1.VT1.2	Ma.2	Mb.1	Mc.2	Md.1	
S1.VT1.3	Ma.2	Mb.2	Mc.3	Md.1	Me.1
S1.VT1.4	Ma.3	Mb.2		Me.2	Mf.1

In the vendor management data managed by the software vendor according to this method, the information of the software configuration is as follows, for example.

```
list of provided object software: S1
software name: S1
list of version type names of software: VT1
list of version number of software: 1, 2, 3, 4
date of version configuration of software: 931005, 931108, 931227, 940131
location of software library: home/S1
Home directory of source program: home/S1/VT1
list of module configuration of source program of each version:
S1.VT1.1 = (Ma.1, Mb.1, Mc.1, Md.1),
S1.VT1.2 = (Ma.2, Mb.1, Mc.2, Md.1),
S1.VT1.3 = (Ma.2, Mb.2, Mc.3, Md.1, Me.1),
S1.VT1.4 = (Ma.3, Mb.2, Md.1, Me.2, Mf.1)
```

To manage an object code program as well as a source code program of software, a source code module name is followed by .s (for example, Ma.1.s) as an extension, and an object code module name is followed by .o (for example, Ma.1.o). The associated, document of the software is divided into modules appropriately, and managed in versions such that a set of documents can be referred to corresponding to each version of the software.

(a2) Management of Software in User Computer

When the above described object software is provided for a user, the easiest method of managing the object software is to store and use the latest version of the object software in the user computer. In the user management data of each user, the information of the configuration of the object software can comprise as follows.

If a user keeps version 1;

```
a list of object software names and version names: S1 (S1.VT1.1), software of other vendors, etc.
software name: S1
software version type name: VT1
software version number: 1
software version configuration date: 931005
software location directory: home/S1
source program location directory: home/S1
source program module configuration: Ma.1, Mb.1, Mc.1, Md.1
```

If a user keeps version 3;

```
a list of object software names and version names: S1 (S1.VT1.3), software of other vendors, etc.
software name: S1
software version type name: VT1
software version number: 3
software version configuration date: 931227
software location directory: home/S1
source program location directory: home/S1
source program module configuration: Ma.2, Mb.2, Mc.3, Md.1, Me.1
```

The body of the object software is stored in modules just as specified in the above described user management data. As shown in this example; if source-code programs are provided, they are compiled in the user computer into object-code modules and are stored together. If only object-

code modules are provided, then the object-code modules are managed in a way similar to the source-code programs in the above example.

(a3) Update Inquiry and Update Instruction

With these settings, the automatic update of the software can be realized as follows.

The easiest method is to indicate the version number of the software assigned by the vendor when the client program of a user computer issues an update inquiry (in a CheckUpdate message). In the example above, the version name consists of the software name, version type name, and version number (for example, S1.VT1.1, S1.VT1.3, etc.). In

this case, the update inquiry message is composed as follows (detailed user identification information is omitted here)

To: Vendor-S1@Vendor.co.jp

From: User1@chem.sci.tokyo-u.ac.jp

Subject: sdmp: CheckUpdate S1

Message:

Software: (Current-Version: S1.VT1.1)

Upon receipt of the message, the vendor's server program (SP) recognizes that the user's current version name (S1.VT1.1) is different from the vendor's latest version name (S1.VT1.4), and edits the information for update. To attain this, the information of module configuration of the versions stored in the vendor management data are used to obtain the differences.

```
S1.VT1.4 = (Ma.3, Mb.1, Mc.1)
(-) S1.VT1.1 = (Ma.1, Mb.1, Mc.1, Md.1),
-----
S1.VT1.4 ← S1.VT1.1 = delete (Ma.1, Mb.1, Me.1)
                        add (Ma.3, Mb.2, Me.2, Mf.1)
```

Only modules requiring additional data are transmitted to the user together with the difference information as update instruction information. In this example, an update command to be returned is written as follows.

To: User1chem.sci.tokyo-u.ac.jp

From: Vendor-S1@Vendor.co.jp Subject: SDMP: Update S1

Message:

Software: (Current-Version: S1.VT1.1)

Updated-Software:

(Description: "A bug was fixed and two new functions are added.")

Version-Type: VT1, Version-Number: 4, Version-Name: S1.VT1.4

Version-Date: 940131

Modules: (Ma.3, Mb.2, Md.1, Me.2, Mf.1)

Update-Direction: (Delete: (Ma.1, Mb.1, Me.1),

Add: (Ma.3, Mb.2, Me.2, Mf.1))

Number-of-Files-Attached: 4)
Treatment-of-Old-Software: Delete
Files-Attached:

source program file of module Ma.3
source program file of module Mb.2
source program file of module Mc.2
source program file of module Mf.1

Upon receipt of the update message, the user's client program interprets the message, and by referring to the user management data, it updates the object software, and then updates the user management data itself. As a result of the update, the new user management data of the user becomes as follows.

a list of object software names and version names:	Sl (S1.VT1.4), software of other vendors, etc.
software name:	S1
software version type name:	VT1
software version number:	4
software version configuration date:	940131
software location directory:	home/S1
source program location directory:	home/S1
source program module configuration:	Ma.3, Mb.2, Md.1, Me.2, Mf.1

According to the update instruction information in the received message, the client program by invoking the update receiving procedure updates the modules of the software using received module files. That is in the above example, three modules Ma.1, Mb.1, and Mc.1 are deleted from the current modules and newly added are four modules Ma.3, Mb.2, Me.2, and Mf.1.

In this case, the old version of the software is not stored. (a4) Update Inquiry and Update Instruction (Another Method)

The method of (a3) is based on the vendor's storage of the module configuration information for all the versions, and users only have to enter the current version names. An alternative method is to send from a user to the vendor the current module configuration information independently of the version name (or as a complement of version name information).

Corresponding to the example above, an update inquiry message from a user can be composed as follows (detailed information such as user identification information is omitted).

To: Vendor-S1@Vendor.co.jp
From: User1@chem.sci.tokyo-u.ac.jp
Subject: SDMP: CheckUpdate S1
Message:
Software: (Current-Version: S1.VT1.1)
Current-Modules: (Ma.1, Mb.1, Mc.1, Md.1)

Upon receipt of the message, the vendor's server program (SP) compares the set of the user's current modules with the latest version set of vendor's modules to obtain the differences.

latest version = (Ma.3, Mb.2, Md.1, Me.2, Mf.1)
(-) user's version = (Ma.1, Mb.1, Mc.1, Md.1)

latest - user's = delete (Ma.1, Mb.1, Mc.1), add (Ma.3, Mb.2, Me.2, Mf.1)

Subsequently, an update instruction message is generated and sent by the vendor, and on receiving the software is updated by the user by the same method as (a3).

(b) Simple Practical Embodiment of Configuration of Software and Generation of Update Information (2)

An example of a little more complicated configuration of software is considered next. Object software S2 has two version types VTa and VTb, and each version type has a plurality of modules and shares a part of them with each other. To properly maintain and manage the software library, the vendor stores all the old versions of modules as well as the latest versions.

(b1) Configuration and Management of Software in Vendor Computer

In the vendor computer, the software library SL of the software S2 can be stored in a single directory (for example, S2.VTab) for the two version types together. Each module is identified by a module name and a module version number (for example, Ma.1, Mb.3, etc.) as in (a).

Corresponding to the two version types, two series of versions are generated, and the software library is required to store the version names and the module configurations of respective version types in a time series as follows.

date of	version type VTa		version type VTb	
	version name	module configuration	version name	module configuration
931005	S2.VTa.1	Ma.1 Mc.1	S2.VTb.1	Mb.1 Mc.1
931108	S2.VTa.2	Ma.2 Mc.2	S2.VTb.2	Mb.1 Mc.2
931227	S2.VTa.2	Ma.2 Mc.2	S2.VTb.3	Mb.2 Mc.3 Me.1
940131	S2.VTa.3	Ma.3 Mc.2	S2.VTb.4	Mb.2 Mc.2 Mf.1

The software library also stores the old version of modules for maintenance (normally the software library packs the entire data to store only the original version of each module and the difference data. Therefore, the entire stored modules (in a restorable packed format) are the following 11 modules.

Ma.1, Ma.2, Ma.3, Mb.1, Mb.2, Mc.1, Mc.2, Mc.3, Me.1, Me.2, and Mf.1

Modules are used either exclusively in a specific version type or commonly among a plurality of version types.

In the vendor management data, the information about the software configuration can be stored as follows.

a list of object software name:	S2
software name:	S2
list of software version type names:	VTa, VTb
list of software version numbers:	(VTa) 1, 2, 3 (VTb) 1, 2, 3, 4
software version configuration date:	(VTa) 931005, 931108, 940131 (VTb) 931005, 931008, 931227, 940131
location of software library:	home/S2
source program location directory:	(VTa) home/S2/VTab (VTb) home/S2/VTab
list of source program module configurations of each version:	(VTa) (VTb)
	S2.VTa.1 = (Ma.1, Mc.1), S2.VTa.2 = (Ma.2, Mc.2), S2.VTa.3 = (Ma.3, Mc.2), S2.VTb.1 = (Mb.1, Mc.1), S2.VTb.2 = (Mb.1, Mc.2), S2.VTb.3 = (Mb.2, Mc.3, Me.1), S2.VTb.4 = (Mb.2, Me.2, Mf.1)

(b2) Dependency on Applicable Environment and Selection of Version Type

In a set of software, a plurality of version types are required (or appropriate) in the following cases.

different software functions in the same applicable environment (various functions)

a number of new function added to the original software in the same applicable environment

different capacities of main storage in the same applicable computer model, OS, and software environment

different applicable software environment (for example, input/output interface software, database interface, runtime library, etc.) in the same applicable computer model, OS, etc.

different OS (for example, due to level up) in the same applicable computer model

hardware-dependency due to different applicable computer model in the same OS

different applicable computer model and OS

The software vendor provides a plurality of version types to properly adopt the software to these differences and is requested to select and provide an appropriate version type depending on the situations of each user. To attain this, the vendor clearly describes and informs the users of the applicable range of each version type, incorporates selection logic in the vendor management data and the vendor procedures and obtains from the users enough information enough to select appropriate version types. For example, the following vendor management data are set for the two version types for (b1).

applicable computer model for software:

(VTa) A100, F3, H20, S1000

(VTb) A200, F3, F4, S1000, S1500

applicable OS for software:

(VTa) OS-A.1.3, OS-F3.2, OS-S.3.1

(VTb) OS-A.1.4, OS-F3.2, OS-F3.3, OS-S.3.3

types and conditions of applicable environment for software:

(VTa) Graphics-1, Memory ≥ 4 MB

(VTb) Graphics-1 or Graphics-2, Memory ≥ 8 MB

The vendor procedure refers to these data to determine an applicable or an appropriate version type for each user.

(b3) Software Management in User Computer

Normally, users only have to store the latest version of an appropriate version type for their environment. The configuration information object software in the user management data is almost the same as that in (a2). For example:

if a user stores version 2 of version type VTa:

a list of object software names and version names:	S2 (S2.VTb.2), software of other vendors, etc.
software name:	S2
software version type name:	VTa
software version number:	2
software version configuration date:	931108
software location directory:	home/S2
source program location directory:	home/S2
source program module configuration:	Ma.2, Mc.2
applicable computer:	A100
main storage capacity:	10 MB
OS:	OS-A.1.3
environment software:	Graphics-1

-continued

If another user stores version 4 of version type VTb:

5 a list of object software names and version names:	S2 (S2.VTb.4), software of other vendors, etc.
software name:	S2
software version type name:	VTb
software version number:	4
10 software version configuration date:	940131
software location directory:	home/S2
source program location directory:	home/S2
source program module configuration:	Ma.2, Me.2, Mf.1
applicable computer:	F4
main storage capacity:	20 MB
15 OS:	OS-F3.3
environment software:	Graphics-1

(b4) Update Inquiry and Update Instruction of Software

The client program (CP) of a user has to perform almost the same processes as (a3) in an update inquiry for the software. For example, for the above described user who stores the version S2.VTb.2, the update inquiry message is generated as follows. (The user environment information is omitted in (a3), but is added below.)

25 To: Vendor-S2@Vendor.co.jp

From: UserA@docs.scit.it.ac.jp

Subject: SDMP: CheckUpdate S2

Message:

Software: (Current-Version: S2.VTb.2)

User: (User-Machine: A100, Main-Memory: 10 MB,

OS: OS-A.1.3, Software-Environment: Graphics-1)

Upon receipt of the message, the server program (SP) of the vendor checks the environment of the user to confirm that an appropriate version type is used by the user. Then, it is determined whether or not the version maintained by the user is the latest version of the version type. If not, the difference between the modules is obtained as in (a3) or (a4). latest version of type VTa S2.VTb.3 = (Ma.3, Me.2,

latest version of type VTb S2.VTb.3 = (Ma.3, Me.2)

(-) user's current version S2.VTb.2 = (Ma.2, Me.2)

S2.VTb.3 ← S2.VTb.2 - delete (Ma.2, Me.2)
add (Ma.3, Me.2)

The server program (SP) returns an Update command message, providing the user with the above described difference information sent as update instruction information together with the body of modules to be added. The user client program (CP) receives the answer message and updates the software S2. All these methods are the same as in (a3).

(b5) Update Beyond Version Type

55 In (b4), software is updated within the same version type. Described below is the method of updating software beyond the version type boundary in the same object software. There are cases in which the software is updated beyond version type. For example:

60 The vendor starts providing the software of a new version type though the user environment remains unchanged (adding functions, developing a level-up version, releasing restrictions, etc.).

Software environment of the user is appropriately improved to solve new problems by using a better version type of the vendor software, though the user computer model and the applicable OS remain unchanged.

A new higher-level OS is introduced in the user computer to obtain a necessary software environment and use a level-up version of the object software, while the user's computer model remains unchanged.

The user has introduced a higher-level computer model and OS, and requests an improved version of the object software.

For example, suppose that the two above mentioned users refer to the same user who switches his or her environment from that of version S2.VtA.2 to that of a new version S2.VtB.4. That is, the user's environment is switched as follows. The changes in the computer model and OS enable the conventional version type VtA to be upgraded into another higher-level version type VtB.

computer model:	A100 → F4
main storage capacity:	10 MB → 20 MB
OS:	OS-A.1.3 → OS-F3.3
environment software:	Graphics-1 → Graphics-1

With the change in the computer model and OS, the user expects an updated and higher-level version of the software. Thus, the user enters an update inquiry command in the client program (CP) to edit his or her user identification information with the interactive editing function, and sends the following update inquiry message.

To: Vendor-S2@Vendor.co.jp
From: UserA@docs.sci.tit.ac.jp
Subject: SDMP: CheckUpdate S2

Message:

Software: (Current-Version: S2.VtA.2)

User: (User-Machine: F4, Main-Memory: 20 MB,
OS: OS-F3.3, Environment-Software: Graphics-1)

Upon receipt of the message, the server program (SP) of the vendor checks the environment of the user to confirm that the user environment has been changed from that recorded in the vendor management data, and then updates the contents of the vendor management data. In the new user environment, it is confirmed that a higher-level version type VtB is applicable. The switching of the version type is automatically performed as follows (assuming that the switching of the version type does not demand extra payment).

The server program (SP) obtains the difference from the user's current version to the latest version of the new version type VtB.

latest version of type VtB S2.VtB.4 ← (Mb.2, Mc.2, Mf.1)
(-) user's current version S2.VtA.2 = (Ma.2, Mc.2)

S2.VtB.4 ← S2.VtA.2 = delete (Ma.2, Mc.2)
add (Mb.2, Mc.2, Mf.1)

The update message in return is sent as follows.

To: UserA@docs.sci.tit.ac.jp
From: Vendor-S2@Vendor.co.jp
Subject: SDMP: Update S2

Message:

Software: (Current-Version: S2.VtA.2)

Updated-Software:

(Description: "Due to your environment change, Version S2.VtB.4 is now installed.

New features are . . ."

Version-Type: VtB, Version-Number: 4,

Version-Name: S2.VtB.4

Version-Date: 940131

Modules: (Mb.2, Mc.2, Mf.1)

Update-Direction: (Delete: (Ma.2, Mc.2),

Add: (Mb.2, Mc.2, Mf.1))

Number-of-Files-Attached: 3

Treatment-of-Old-Software: Keep for one month

Files-Attached:

source program file of module Me.2

source program file of module Mc.2

source program file of module Mf.1

Upon receipt of the update message, the user's client program interprets the message, refers to the user management data, updates the user management data, and then updates the object software to generate a new version S2.VtB.4. Since the change in the object software is significant, the old version of the software (version S2.VtA.2) is not deleted completely, but is allowed to keep and use for a back-up purpose for one month as indicated by the update message. As a result of the update, the user management data of the user reads as follows.

a list of object software names and version names: S2 (S2.VtB.4, S2.VtB.2), software of other vendors, etc.

(for version S2.VtB.4:)

software name:	S2
software version type name:	VtB
software version number:	4
software version configuration date:	940131
software location directory:	home/S2
source program location directory:	home/S2
source program module configuration:	Mb.2, Mc.2, Mf.1
computer model:	F4
main storage capacity:	20 MB
OS:	OS-F3.3
environment software:	Graphics-1
(for version S2.VtB.2:)	
software name:	S2
software version type name:	VtB
software version number:	2
software version configuration date:	931108
software location directory:	home/S2/old
source program location directory:	home/S2/old
source program module configuration:	Ma.2, Mc.2
computer model:	A100
main storage capacity:	10 MB
OS:	OS-A.1.3
environment software:	Graphics-1

(Note: With the old version, the new computer will not work as is. If the data are stored in the shared file server of the user computer and the original computer model A100 can be connected when necessary, then the old version software can still be useful.)

(b6) Update Beyond Version Type (2) Providing New Services

The process in (b5) above refers to the update beyond version type, but is automatically performed under the situation that the update process does not impose extra charge on the user. On the other hand, there are cases when the update beyond version type refers to an upgrade to a new service which requires the user to pay extra charge. In this case, the server program returns a Notice message to prompt the user for his or her decision. If the user requests the purchase of the new service, then the user sends a WantRenew message. In this case, the user and the vendor communicate messages and perform the necessary processes as follows.

As in (b5), the user sends the CheckUpdate message with the change in the computer model and OS. The server

program recognizes that the version type required for the model and OS is a newly served version imposing an extra charge on the user. Thus, it returns the following Notice message.

To: UserA@docs.sci.tit.ac.jp

From: Vendor-S2@Vendor.co.jp

Subject: SDMP: Notice S2

Message:

Software: (Current-version: S2.VTb.2)

Notice: "Due to your environment change, a new-service version S2.VTb.4 is to be used. Please follow the renewing procedure described in the attached file, and issue a 'WantRenew' message."

Number-of-Files-Attached: 1

Files-Attached:

Upgrading the software from version S2.VTa to version S2.VTb

(A document file describing new functions, applicable environment, purchase procedure, and WantRenew command)

Upon receipt of the Notice command, the client program first stores the command and displays the message to the user. The user reads the contents of the message and document files. After following the procedures of the payment for the purchase, the following WantRenew command is issued.

To: Vendor-S2@Vendor.co.jp

From: UserA@docs.sci.tit.ac.jp

Subject: SDMP: WantRenew S2

Message:

Software: (Version-type: VTb,

Current-Version: S2.VTb.2)

User: (User-Machine: F4, Main-Memory: 20 MB,

OS: OS-F3.3, Environment-Software: Graphics-1)

(c) Practical Example of Fault Report System

An example of an automatic fault report was explained as the first embodiment by referring to FIG. 7.

The following description complements the explanation.

When the object software terminates abnormally, the operating system (OS) normally detects the abnormal termination and find out the cause of the abnormal termination (that is, no "/bin/play" command to be executed is found) and the source of the cause ("ISISInfo").

The system should be set such that the client program can be automatically by the OS activated and the above information is automatically informed of when an abnormal termination occurs.

To trace back the executed instructions from the one which directly causes an abnormal termination. It is a function of a debugger in a programming language and to locate the source codes of the sequence of the instructions.

There are not many language systems which provide this function in a form available to other programs, but Lisp and tcl/tk are the examples of having such a function. The example is obtained using the tracing function of the tcl/tk language for the object software written in tcl/tk.

A fault report message is generated in the format of normal electronic mail and sent to the server program through the network.

As described above, the software distribution/maintenance system according to the second embodiment of the present invention is a generalizing extension of the software distribution/maintenance system according to the first embodiment of the present invention.

In the first embodiment, a single set of object software is to be distributed and maintained. Thus, a user who uses a

number of sets of software provided by different software vendors needs to be served by separate and different software vendors needs to be served by separate and different software distribution/maintenance systems. For improving this situation on the user side, the software distribution/maintenance system according to the second embodiment of the present invention is built to handle a plurality of sets of object software in a unified way by overcoming the problems of variations.

For the purpose of improving this situation on the user side, the software distribution/maintenance system according to the second embodiment of the present invention is built to handle a plurality of sets of object software in a unified way. Thus, from the viewpoint of a user, it can be regarded as a one-user * many-(one-software-one-vendor) system. By combining the viewpoints of users and vendors, the system can be regarded as a many-user * many-(one-software-one-vendor) system.

In a wider viewpoint, this system certainly accepts the situation that one set of object software having a number of version types are developed and distributed by a number of vendors and are used on various types of platforms. Thus, the software distribution/maintenance system according to the second embodiment of the present invention forms a unified system of a many-user * many-software * many-vendor * many-platform type.

Since the networks of computers are already built and operated in global scales, it is naturally expected that the software distribution/maintenance system according to the second embodiment of the present invention can serve as a unified infrastructure system for the distribution and maintenance of software in the world-wide scale, if its protocol is internationally standardized. Thus the system and method according to the present invention are expected to have innovative effects on the systems and methods of software distribution and maintenance, in a similar way that the electronic mail system and method had innovative effects on the mailing systems and methods.

As described above, the software distribution/maintenance system according to the second embodiment of the present invention is a generalizing extension of the software distribution/maintenance system according to the first embodiment of the present invention. In the first embodiment, a single set of object software is to be distributed and maintained. Thus, a user who uses a number of sets of software provided by different software vendors needs to be served by separate and different software distribution/maintenance systems. For improving this situation on the user side, the software distribution/maintenance system according to the second embodiment of the present invention is built to handle a plurality of sets of object software in a unified way by overcoming the problems of variations.

The effects of the second embodiment of the present invention are summarized below briefly:

Concerning to one set of software distributed and maintained by a vendor for a large number of users, the effects of the second embodiment of the present invention include all the effects of the first embodiment of the present invention. That is, for maintaining the object software in a user computer, an update inquiry message is automatically sent by the client program to the vendor on the simple activation either by the user's command input, by the activation of the Object software, or by the instruction of user's software such as a daemon program; as the answer to it, an update instruction message is automatically composed and returned by the vendor's server program to the user; and then, on receiving the answer, the client program automatically

updates the object software. A fault report is also automatically sent to the vendor on the occasions of abnormal user/vendor procedures for specific handling of different types of messages.

Such standardized structure of the server/client programs has an important effect to standardize the protocols for the software distribution/maintenance. (This point may be compared to the standardized structure of the electronic mail program, which can handle messages with standardized header and arbitrary contents.) The second embodiment of the present invention thus can form a basis of such standardized protocols.

In concert with the growth of networks in the global scales, the software distribution and maintenance system according to the second embodiment of the present invention is expected to make a worldwide infrastructure system for software distribution/maintenance. The effect of such a system gives innovative impacts on software industries, user industries, and whole range of users, organization and individual.

According to the technique of the second embodiment, a method to resolve the problems 1 through 4 is proposed. However, a new problem arises in the second embodiment, which also existed implicitly in the first embodiment.

As shown in FIG. 8, a client program is expected to handle a number of object software systems of one or more users in the second embodiment. For the client program, for example, it is required to design to handle a plurality of pieces of object software for a plurality of users. Therefore, it is complicated. Especially, an interface to call the user process procedure (UP), and handling of parallel processing when a user requests a plurality of processes are difficult. The similar problem may arise in the server program SP.

The object of the third embodiment is to distribute software on a worldwide scale, or build a standardized technical framework suitable for a form of software distribution without making the designs of the client program CP and the server program SP complicated. That is, the present invention is intended to provide software distribution and maintenance system and method, which uses a plurality of and a variety of pieces of software as objects and can be utilized by a plurality of and a variety of software users linked on a worldwide scale over a communications network between computers, and comprises a capability to rapidly and properly acquire and use software created and updated by software vendor.

FIG. 18 is a block diagram showing a principle of the present invention. In this figure, 81 through 83 are user computers, 84 and 85 are vendor computers, and 86 is a general-purpose network for connecting the user computers and the vendor computers.

The user computers 81 is used by a user U1. The user computer 82 is used by users U2 and U3. The user computer 83 is used by a user 4. The vendor computers 84 and 85 are generally each used by one software vendor, and are respectively used by the vendors V1 and V2. Generally, one or more object software applications (software applications as objects for distribution and maintenance, which generally correspond to one or more users) are installed in the user computers. Management data for user UD and the first process program CP, for example, are installed for each object software application.

On the user computer 82, for example, object software applications S2.b and S1.c are installed for the user U2, management data for user and the client program CP are installed for each object software, and object software S2.a, the management data for user U2, and the client program CP

are installed for the user U3. S1 and S2, for example, distinguish object software applications, and a, b, and c indicate their versions or (version types).

The first process program, the client program (CP), for example, handles only one piece of object software for one user. It uses dedicated management data for users UD and user process procedures (UPs), to be described later, to distribute and maintain the object software. That is, it sends various messages such as a software purchase request made by a user to a vendor, an update inquiry, a fault report, etc., made by a user to a vendor and performs processes such as an update of user software etc. after receiving an answer message from the vendor. When sending and receiving messages, the client program executes these processes based on standardized protocols of the software distribution/maintenance system.

The management data for user UD, consisting of user identification information, vendor identification information, software management information, etc., is used when processes such as sending messages, updating user software, etc. are performed by the client program CP.

The vendor computers 84 and 85 are each utilized by one vendor. Inside the vendor computer 4, for example, software libraries SL (S1) and SL (S2) are installed, and corresponding management data for vendor VD and the second process program, the server program SP, for example are installed for each library. The software library SL (S1) generally consists of a plurality of versions (or version types) of the object software S1. For example, the software library SL (S3) in the vendor computer 85 consists of two versions (or version types) S3.a and S3.b of the object software S3.

The management data for vendor VD, consisting of information such as vendor identification information, user management information, software management information, etc., is utilized by, for example, the server program SP.

The server program SP handles only one object software library for one vendor, similar to the client program CP installed on the user computer, and performs distribution/maintenance of the object software using the dedicated management data for vendor VD and vendor process procedures (VP), to be described later.

A network 86 is a general-purpose network, which allows message communications between the client program CP installed on the user computer and the server program SP installed on the vendor computer. By standardizing protocols, messages of the software distribution/maintenance system can be sent/received between the user and the vendor. The network 86 may be connected all the time, or connected depending on need similar to an electronic mail connection to send and receive messages.

In FIG. 18, the user computer and the vendor computer are generally different to each other. Some of the computers are, however, used by a user serving as both a user and a vendor. Therefore, one computer may sometimes include both the client program CP and the server program SP. In this case, the client program CP and the server program SP behave in exactly the same manner as they behave individually.

As described above, the vendor manages a number of versions of one piece of software to provide them to a number of users. The user takes advantage of one version of each software and a number of types of software (provided by different vendors) at the same time. Thus, the present invention is based on the assumption that a number of versions and types of the object software are provided by a number of vendors and used by a number of users.

Further explanation on the principle of the present invention is given below, referring to the FIG. 18. The software distribution and maintenance system using a network that a number of users U1, U2, . . . using a number of types of object software S1, S2, . . . to be distributed, managed, and maintained, and a number of software vendors supplying the above object software manage the distribution and maintenance of the above object software over a computer network, comprising:

- (a) one or more first process programs, CPs, installed in each of user computers 81 through 83, that manage object software groups S1, S2, . . . to be used by the users U1, U2, . . . individually for every object software and for every user using the object software;
- (b) one or more second process programs SPs, installed in either of software vendor computers 84 or 85, that gives services to vendor software libraries SL(S1), SL(S2), . . . for each of the software libraries;
- (c) a network 6 that connects the first process program CP installed in the user computer to the second process Oprogram SP installed in the vendor computer, based on standardized communications protocols;
- (d) a capability that performs distribution and/or maintenance for the above object software by sending a message that requests to distribute and/or maintain the object software for one piece of object software via the network 86, according to instructions given by the users U1, U2, . . . or a user-defined program, receiving an answer message from the second process program SP, and depending on the contents of the answer message and settings made by the user U1, U2, . . . ; and
- (e) the second process program SP that has a capability to reference the software libraries SL(S1), SL(S2), . . . managed by the vendors V1, V2, . . . depending on the contents of a received message and settings made by the vendor V1, V2, . . . for the object software specified with the message when receiving the above message from an arbitrary first process program CP, to generate an answer message to request distribution and/or maintenance of the above object software, and to send the answer message to the first process program CP of the sender of the corresponding message.

Next, the software distribution and maintenance method according to the present invention is explained below.

In FIG. 18, the users U1, U2, . . . specify desired software (object software) and its vendors V1, V2, . . . and invoke an acquisition inquiry instruction, the first process program CP then sends a purchase inquiry message to the software vendors V1, V2, . . .

The second process program SP, corresponding to the object software library installed on the software vendor side, receives this message, verifies a qualification of the user, automatically performs an appropriate process depending on settings of the management data for vendor VD, and returns an answer message to the user.

The first process program CP that sends the purchase inquiry message receives the answer and advises the user. If the software itself is returned, the first process program CP stores it according to instructions included in the answer message.

Then, the first process program CP references the management data for user UD, and performs required compilation and linking, etc. based on settings of that data so that the software can automatically be installed in a pre-specified area.

Additionally, the first process program CP makes an inquiry about whether or not an update or improvement is

made for software currently used by the users U1, U2, . . . If the software is updated, the users U1, U2, . . . specify a desired software name to be updated and requested, and invoke the first process program CP. After the first process program CP references the management data for user UD to extract a configuration of the software Si.v that the user is currently using, it sends an automatic update inquiry message with the user identification information appended to the vendors V1, V2, . . .

When a corresponding second process program SP receives this message, it verifies a qualification of the user according to the settings of the management data for vendor VD, and returns update information about the software. To a qualified user, a message of "no update" is returned as update information such as when the software is not updated. If the software is updated, messages of "an update method for a user side" and "the software itself for an update" are returned.

When the above reply is received, the first process program CP firstly stores, for example, the updated Osoftware, according to the settings of the management data for the user, replaces a previous version with the updated version according to update instructions received, etc. It performs compilation and linking if needed, to allow execution of the updated version. As an invocation method for an update inquiry, an invocation by a user, an automatic invocation when the user calls a software, an invocation at a specified time (for example, the start of every day, every week, every month, etc) is made by a demon program.

If software installed on the user computer managed by the first process program CP abnormally terminates while running, the first process program detects a cause and a state of the fault and automatically sends a fault report to the vendor V1, V2, . . .

A corresponding second process program SP registers this and reports it to a developer. In addition, it returns a message acknowledging a receipt of the message to the user.

After software bugs are manually fixed by the developer, the software is registered in an object software library of the vendor. After that, the update version is provided to the user as the latest version.

Further explanation on the software distribution and maintenance method implemented with the present invention is given below with reference to the FIG. 18.

The above software distribution and maintenance system performs the following processes:

- (a) Each user U1, U2, . . . invokes a first process program CP corresponding to one piece of object software by entering a command on the user computer U1, U2, . . . or by invoking a command from a user-specified program;
- (b) the first process program CP sends a message to request distribution and/or maintenance of the object software to a corresponding second process program Sp via a network 86;
- (c) the second process program SP receives the message from the above first process program CP on the vendor computer for the above object software;
- (d) references the software library managed by the vendor V1, V2 based on the contents of the received message and the settings of the object software specified by the message, generates an answer message to request distribution and maintenance of the above object Osoftware, and returns the above answer message to the first process program CP of the user that sent the corresponding message via the network 86.
- (e) The first process program CP receives an answer message from the above second process program SP on the

user computer U1, U2, . . . in the sender side of the above message, and performs the distribution and/or maintenance processes for the above object software based on the contents of the reply and settings of the user U1, U2, . . .

As described above, the present invention installs the first process program, for example, the client program, on the user computer for every object software and for every user. The invention also installs the second process program, for example, the server program SP, for every software library on the vendor computer. By executing a message transfer between the client program CP and the server program SP based on standardized protocols, a standardized software distribution and maintenance system and method can be implemented on a very large scale, for example, on a worldwide scale though the method is primarily proposed for each software.

Description of the Preferred Embodiments

(1) System Structure and the Network

FIG. 19 is a block diagram showing an embodiment of the software distribution and maintenance system that takes advantage of a communications network implemented with the present invention. In this figure, a user computer 91, a vendor computer 93, and a user/vendor computer 94 are connected over a network 92. Compared with FIG. 18, that shows a principle of the present invention, there is a basic difference in FIG. 19 that the user/vendor computer 94 is used by a user serving both as a user computer and a vendor computer for software. The user/vendor computer 94 is equipped with a client program CP for the software to be used as a user, and a server program SP for a software library SL (S3) to be provided as a vendor.

Further compared with FIG. 18, there is another difference in FIG. 19 that a user process procedure group (UP) corresponding to each client program CP, and a server process procedure group (VP) corresponding to a server program (SP) are added. Details of these groups are described later.

The present invention comprises a capability to send and receive messages between the client program CP installed in the user computer and the server program SP installed in the vendor computer. Detailed explanation of the above capability is given below:

This invention is based on the assumption that message communications can be performed between the client program (CP) of a certain user managing one piece of object software, and the server program (SP), managing a corresponding software library of the vendor supplying the object software. Such communications capability is one of basic capabilities of a network.

A specific name can be used for a process that performs a group of operations on each computer (the above CP and SP can be in a form of processes) under an operating system (OS). With this technique implemented (on a user computer, for example), a specific process name can be created for each client program (CP), which should be composed of the combination of a client program name of the system of the present invention, the user name, and the object software name.

Meantime, a network system (the internet system, for example), recognizes a process on each computer, and generally provides transmission from each process to outside, and reception from outside computers to each process. If a network address of an other computer connected to the network is given, a message can be sent to that computer.

Accordingly, by specifying a network address computer and a name of process in the receiver side, and by repre-

senting a message based on an appropriate network communications protocol, a message can be sent from a process installed in one computer (a client program, for example) to a process installed in another computer (a server program, for example) by taking advantage of the capabilities of the operating system (OS) and the network system. Thus, peer-to-peer communication can be implemented between the client program and the server program.

Of course, a main process that handles communications made by a plurality of client programs (CP) on each computer can be set. In this case, the main process handles message transmission and reception processes in parallel. Processing of distribution and parallel processing are required in this case. A destination address of the communication message is represented by the network address of the partner computer and the main process is represented as a partner process.

However, a message processing (especially, a parallel processing and a distribution processing) performed in the main process, is similar to a basic processing performed in a communications network. Thus, it is not a matter whether such processing is performed in the main process or in the general-purpose network. (Furthermore, it is an intention of the present invention to let the general-purpose operating system (OS) and the network system perform such processes as described above, instead of the main process that explicitly performs them.)

As described above, when process groups exist in each computer, communications between peer to peer can be performed as well as communications between group to group by use of established techniques.

According to the object of the present invention, a communications method implemented with the present invention is simply expressed as a method of "communications between processes over a general-purpose network", which may be either in peer to peer or in group to group performed in a ratio of 1 to 1 and a ratio of group to group.

(2) A Function of the System of the Third Embodiment

This function is similar to that of second embodiment and its explanation is abbreviated.

(3) Kind and Structure of Messages

The kind and function of the messages and the command set used in the third embodiment are similar to those of the second embodiment, and the explanation is also abbreviated. The structure and syntaxes of the messages in the third embodiment are described below.

(3.1) Structure of 'Upward Messages' (from Users to Vendors)

The 'upward messages' from users to vendors have the following structure:

(header part)

receiver:	vendor's name and network address
sender:	user's name and network address
command line:	SDM system name, SDM command name, and name of the target software
in reply to:	reference to the vendor's prior message (exceptional)

(attributes part)

user information:	information to qualify the user for the access
vendor information:	information of the current version of the target software

-continued

environment information:	information of user's hardware/software environments
<u>(instruction part)</u>	
instruction:	specifying information of the inquiry/request, optional and dependent on the command

(3.2) Structure of 'Downward Messages' (from Vendors to Users)

The 'downward messages' from vendors to users have the following structure:

<u>(header part)</u>	
receiver:	user's name and network address
sender:	vendor's name and network address
command line:	SDM system name, SDM command name, and name of the target software
in reply to:	reference to the user's inquiry/request message
<u>(attributes part)</u>	
user information:	information to confirm the user
vendor information:	(detailed) information about the vendor
software information:	information of the new/updated version of the target software
environment information:	requirements of user's hardware/software environments
<u>(instruction part)</u>	
instruction:	details of the reply for distribution/maintenance and instruction to users, especially the instructions of installation or updating optional depending on the command type.
<u>(module part)</u>	
module:	the body of new/updated modules to be delivered

(3.3) Syntax of the Messages

The syntax of the upward and downward messages is described here in the human readable textual form. Actual coding of messages on the network may depend on the network system.

The specifications of the receiver, the sender, and the command line in the header part of the message are always necessary. They are written line by line in the form of keyword-and-value (or value list) pairs using a delimiter such as a colon in between.

The attributes part is similarly written with lines of a keyword-and-value pair. In case the specification of the value is complex, the complex value may be written by a group of keyword-and-value lines enclosed in parentheses. In a manner similar to programming languages, we may define various delimiters: such as parentheses and semicolons for explicitly define a list of values (these may be replaced by spaces in an implicit but clear case of a list of values), a CR for a line break, a '/' mark (or Yen mark in Japanese) for line continuation, etc. Thus, it may be described in the following manner:

keyword 1:	value 1
keyword 2:	(keyword 21: value 21)
	keyword 22: value 22; value 22; value 223)
	keyword 23: value 23)

As the topmost keywords of the attributes part, we choose the four keywords of 'user', 'vendor', 'software', and 'environments'.

The instruction part may also be written in the form of keyword-and-value lines. For example, the instructions to add modules and to delete modules are written in this form. The keywords to be used in the instruction part may depend on the command of the message.

The module part may also be written (within a textual representation) in the form of 'module: (the body of a module)'.

One advantage in specifying the messages in the form of the keyword-and-value pairs is that the specified information may be stored in its original form (or after re-assigning the keyword) in the user management data (UD) or in the vendor management data (VD). Once the system of keywords is clearly defined in their names and semantics, the information stored in the form of keyword-and-value pairs can readily be searched, updated, and analyzed.

(3.4) Hierarchical Structure of Keywords

As described above, the messages are described in a hierarchical scheme of keyword-and-value pairs. A hierarchical system of keywords is chosen in the following way:

<u>(header part)</u>		receiver
	sender	command line
	in reply to	
<u>(attributes part)</u>		
<u>(user information)</u>		name
user:	organization	address
	network address	status
	id & password	
<u>(vendor information)</u>		name
vendor:	organization	address
	network address	status
	id & password	
<u>(software information)</u>		name
software:	version type	current version
	configuration	form
	description	
<u>(environment information)</u>		
<u>environments:</u>		
hardware:	machine	memory
software:	OS	
directory path		
(instruction part)		
instruction:	payment	wasted
	old version treatment	delete
	add	
<u>(module part)</u>		
module		

(4) Process Performed on a User Computer

A client program (CP) is installed on a user computer to manage an acquisition and update of one piece of object

software by one user. This program edits an upward message and sends the edited message via a network when operations such as a request for a new acquisition or an update inquiry, etc. are performed by a user. It receives a reply from a server program (SP) to perform installation, etc. An entire flow-chart of the process is shown in FIG. 23, and each step of the process flow is described below.

Step S71:

A client program (CP) is invoked when a command is entered by a user or by user software (a demon program, etc.), or when a command is entered at an invocation of the object software.

Step S72:

The client program calls a user process procedure to interactively or automatically edit an inquiry/request message when the command is entered. Within the user process procedure, a message is generated by referencing and searching management data for user (UD). Message types are a summary inquiry, a request to newly acquire new software, a request to receive a new service, an update inquiry, a fault report, etc.

Step S73:

The client program registers a message and sends it to a corresponding server program of a vendor via a network, and waits for a reply message. To keep confidentiality for communications, a portion or a whole of the message is sometimes encrypted before being sent.

Step S74:

The client program receives a reply message from the server program (S) of the vendor via the network.

Step S75:

The client program calls the user process procedure for a reply message (UP) P3, and verifies that the reply message is sent from the legal vendor using management data for user (UD). When a portion or a whole of the received message is encrypted, the program decrypts it using an appropriate key to prove that both the user and the vendor are legal.

Step S76:

The client program calls a user process procedure for a reply message reception when receiving the message command. Reception messages are summary information, new supply information, addition information, an update, a non-update, a notice, a warning, a reception of fault report, a comment answer, etc. The user process procedure (UP) analyzes the received message and stores its contents. When determined to be appropriate, installation is performed according to instructions being sent by the vendor to perform processes such as an update of management data for user (UD). These processes reference and update management data for user (UD) and object software.

Step S77:

When an automatic report of a result of a received message process is desired (in case of a new supply, an additional supply, and an updated message), a result of a process performed in Step S76 is monitored. An installation verification message or an installation error message is respectively returned to the server program (SP) of the vendor if the process is properly executed, or if a problem arises. Step S78:

The client program registers the reception of the reply message and a result of the process, returns a control to the process that calls the client program (CF). Then, the program is terminated (enters into an idle state).

The process of the client program (CP) described above basically conforms to those of the first and the second earlier applications. There are, however, the following differences in particular. The difference from the first earlier application

is that the client program (CP) calls the user process procedures (UP) (by adopting the method of the second earlier application), to make it clear that the management data for user (UD) is used. While the difference from the second earlier application is that no sorting or parallel processings are required for a plurality of users and a plurality of types of object software management since the client program (CP) manages one piece of object software for one user, which allows simplification of the processes when receiving an answer message or when entering a command (likewise the first earlier application).

(Note that the parallel and sorting processing can be more generally performed as capabilities of an operating system (OS) and a network due to the existence of a plurality of client programs (CPs). Therefore, it may not be processed within the software distribution and maintenance system of the present invention, and the above simplification can be implemented.)

(5) Process Performed in a Vendor Computer

A server program (SP) is installed on a vendor computer to distribute and maintain object software of each vendor. Inquiry/request messages such as a request for a new software purchase, an update inquiry, etc., are sent from a number of client programs of users over a network. The server program (SP) receives these messages, generates an answer message for services such as a new software supply, update information, etc., and returns them in response to the request made by the client program over the network.

An entire flow of the message process performed by the server program (SP) of the vendor is shown in FIG. 24. Each step in the figure is described below:

Step S81: A server program (SP) installed in a vendor computer is always invoked to wait for receiving various messages of inquiry/request sent by a client program (CP) over a network.

S82: The server program receives a message from a client program (C7) over the network. (Normally, a sub-process is created to process a message performed in steps S83 through S87. The main process of the client program reenters a standby state. There is no distinguishing between the main process and the sub-process for simplicity.)

Step S83: The server program extracts a user name and user information from the messages, and calls a vendor process procedure (VP) for user qualification verification P11. With the user qualification verification VP, user information is verified to determine whether or not the user is qualified to perform an access with this command, using management data for vendor (VD).

Step S84: For an unqualified user, the server program calls an unqualified user process procedure (VP) P12, and generates an appropriate answer message such as a refusal, warning, information about acquiring qualification, etc. After that, the processing goes to Step S86.

Step S85: For a qualified user, the server program calls a vendor process procedure for a reception/answer process (VP) P13 corresponding to a command included in a received message, and generates an answer message to the user. The answer message is created according to the protocols of the software distribution and maintenance system.

The vendor process procedure (VP) analyzes the received message, extracts information from the vendor management data and the object software library to be provided, and generates a message. Additionally, it stores the provided contents, etc. in the user information included in the vendor

management data (VD). There are such answer message types as a summary information supply, a new software supply, a new service supply, a message of update information, a message of no update, a comment answer, etc. There are additional message types such as a Notice message, warning message, etc. as exceptions.

(6) Usage

Basic usage and operation methods of the third embodiment of the present invention are similar to those of the second embodiment; thus the explanation of them is abbreviated. The following way of using the present invention is new and suitable to the third embodiment of the present invention.

Bootstrapping of the Client Program

So far in our description of the embodiments of the present invention, we assume that users already own the software components (i.e., a client program (CP), user procedures (UPs), etc.) of the present software distribution and maintenance system when they want to get a new target software system. Some standard client program (CP) and user procedures (UPs), which we propose in our second embodiment of the present invention, may be obtained earlier in some conventional way such as with a portable media or with ftp. However, for making the software distribution and maintenance system in the present invention better fit to each target software, the CP and UPs should better be designed for the target software by the vendor and be supplied to the users "beforehand". This is particularly intended and desirable in our third embodiments of the present invention.

This problem is readily solved. By the use of the present invention, users can obtain from vendors, with minimal preparation and burden, a new target software system and a set of CP and UPs which best fits for the distribution and maintenance of the target software, in the following steps: Step 1: A user has to prepare a minimal set of CP and UPs which has the function of sending an Inform message for summary information inquiry and of processing the Reply-Info message to be received in reply. We may assume that this minimal set of CP and UPs is provided in public as a standard tool and is easily obtainable in any traditional way of software distribution.

Step 2: By use of the minimal set of CP and UPs, the user now sends an Inform message for obtaining summary information of any target software he wants.

The vendor sends back a ReplyInfo message which contains the summary information of the target software and a basic set of the present software distribution and maintenance system sufficient for newly requesting the target software. The basic set may include: a basic CP, user procedures(UPs) for sending a WantNew message and receiving its reply, and user management data(UD) useful for composing the WantNew message.

Step 3: By means of the basic set of software distribution and maintenance system supplied above, the user may send a WantNew message to the vendor to request the target software.

In response, when the vendor returns a NewInstall message for newly supplying the target software, the vendor provides the user with a full set of the present software distribution and maintenance system which is built best fit to the target software. The full set may contain the whole of the client program (CP), a full set of user procedures (UPs), and instructions to install these CP and UPs.

Step 4: Now the user can use the full set of the user's components of the software distribution and maintenance of the present invention; by using it, he can easily update and

renew the obtained target software in the manner described so far. The method described here may be used further for the vendor of the target software to maintain the software distribution and maintenance system itself at the user's site.

This example of embodiment clearly shows that the software distribution and maintenance system suitable for the target software can be composed on users' computers without any burden of users and that the distribution and maintenance of various target software can be achieved readily by use of the present invention.

One problem of the second embodiment of the present invention is the complexity in constructing the standard client program (CP) which invokes different sets of user procedures(UPs) for handling a variety of object software systems. The interface between the CP and UPs needs careful design and can not be flexible. A solution to this problem has guided us to the third embodiment of the present invention as described above. The solution compromises the ideas of the first and second embodiments and takes advantages of them both. The third embodiment of the present invention adopts the simplicity of the first embodiment by limiting to handle only one target software system, while it adopts the whole ideas of the second embodiment to overcome the variation problems; they include clear separation among the client program (CP), user procedures (UPs), user management data (UD), and target software and the standardization of the message protocols.

Effects of the Third Embodiment

The effects of the third embodiment of the present invention are summarized here briefly:

The choice that the client program (CP) handles a single target software for a single user has released the CP's burden of complexity in the multitask processing of possibly many target software systems and many users. This makes the CP simple to handle one inquiry/request from the user at one time.

The same choice has reduced more significantly the CP's burden of complexity in adapting to the variation in target software and in user. The CP is constrained by the standard message protocols to communicate with the server program SP, but has full flexibility inside. Namely, the CP invokes its own user procedures (UPs) and the UPs access to the user management data (UD) and the target software. Thus, whole set of these CP, UPs, UD, and target software can be designed in a set according to the principles of the present invention; interfaces among them may be chosen flexibly and appropriately for each target software, without being constraint of the choices for other target software.

This has simplified the design of the CP, UPs, and UD, and has made the present invention much feasible and practical for wide variety of target software.

The proposal of message protocols for software distribution and maintenance in the third embodiment of the present invention is also an important step towards establishing such a standard. The message types used in the second embodiment of the present invention are used and the syntaxes of the messages are made clearer by the definition of a hierarchy of keywords to be used in the keyword-and-value presentations of information. Such a definition of protocols needs to be examined further in the practice.

As described so far, the choice of single target software in the third embodiment does not mean to set back the software distribution and maintenance method in the present invention to be useful only for the specified target software. Rather, the choice makes the present invention widely applicable to any software system as its target of service. The client/server systems according to the third embodiment

of the present invention may be constructed and used by various vendors to serve their users. Since the methods for users to use them are standardized as described in the present invention, users can use such software distribution and maintenance systems as if they were a standardized tool without noticing the differences inside them.

Just as described as the effects of the first and second embodiments of the present invention, the effects of the third embodiment of the present invention are expected to be innovative in the distribution and maintenance of software. Once vendors develop and update the software for service on vendors' computers, huge number of user in the global scales can obtain and use the new and corrected versions of software almost immediately on users' computers without burdens of labor and skills in obtaining and installing them. The software distribution and maintenance systems of the present invention take care of the automatic or semi-automatic communication between the users' and vendors' computers through the networks. The users may not notice the users' software systems are corrected automatically in this manner of network service. The rapid growth of the network in the global scale will make the services of software distribution and maintenance according to the present invention useful and indispensable in the global scale.

What is claimed is:

1. A software distribution/maintenance system having a plurality of user computers and a vendor computer connected to the user computers via a network to manage and automatically update over the network a set of object software sent and stored in the user computers from the vendor computer via the network, comprising:

an object software library in the vendor computers;

first process means in the user computers for sending an inquiry with current configuration information of the object software stored in the user computers to the vendor computer via the network to inquire about a latest configuration, for receiving an answer from the vendor computer, and for updating the object software stored in the user computers according to an instruction in the answer; and

second process means in the vendor computer for receiving the inquiry from said first process means in the user computers, for generating update instruction information for the object software to match the current configuration of the object software in the user computers with the latest configuration of an updated version in said object software library stored in the vendor computer, and for returning via the network to said first process means the answer with the update instruction information and the updated version of the object software.

2. The software distribution/maintenance system according to claim 1, wherein said first process means in the user computers automatically informs via the network the vendor computer of an abnormal termination and state if an operation of the object software abnormally terminates in the user computers.

3. The software distribution/maintenance system according to claim 2, wherein said first process means automatically informs the vendor computer of the abnormal termination, an instruction causing the abnormal termination, a reason for the abnormal termination, a series of instructions which invoked the instruction causing the abnormal termination, and an environment of software and hardware used when the abnormal termination is detected.

4. The software distribution/maintenance system according to claim 2 or claim 3, wherein said first process means

can automatically inform, via the networks the vendor computer of the abnormal termination by general-purpose electronic mail.

5. A software distribution/maintenance method for managing and automatically updating a set of object software stored in user computers from a vendor computer via a network, comprising:

sending, by a first process in the user computers, current configuration information of the object software to a second process in the vendor computer via the network with an inquiry for a latest configuration of the object software;

receiving by said second process in the vendor computer the inquiry from said first process;

generating update instruction information for the object software to update a configuration of the object software in the user computers with the latest configuration of an updated version of the object software in an object software library stored in the vendor computers;

returning via the network to said first process an answer with the update instruction information and the updated version of the object software; and

processing by said first process stored in the user computers the answer from said second process to update the object software according to the update instruction information in the answer, including preparing for compiling and linking of programs if necessary.

6. The software distribution/maintenance method according to claim 5, wherein when the object software is activated in the user computers, said first process

immediately sends the current configuration information of the object software to said second process in the vendor computer via the network to inquire about the latest configuration,

receives the answer from said second process, updates the object software according to the update instruction information in the answer, and

prepares for compiling and linking programs if necessary.

7. The software distribution/maintenance method according to claim 5, wherein when said first process is activated in the user computers at a predetermined time, said first process

sends the current configuration information of the object software to said second process in the vendor computer via the network to inquire about the latest configuration,

receives the answer from said second process, updates the object software according to the update instruction information in the answer, and

prepares for compiling and linking of programs if necessary to realize an automatic update of the object software.

8. The software distribution/maintenance method according to claim 5, wherein when a user instructs the user computers to activate said first process, said first process

sends the current configuration information of the object software to said second process in the vendor computer via the network to inquire about the latest configuration,

receives the answer from said second process, updates the object software according to the update instruction information in the answer, and

prepares for compiling and linking of programs if necessary.

9. The software distribution/maintenance method according to claim 5, wherein following processes are interchangeably performed:

when the object software is activated in the user computers, said first process immediately sends the current configuration information of the object software to said second process in the vendor computer via the network to inquire about the latest configurations, receives the answer from said second process, updates the object software according to the update instruction information in the answer, and prepares for compiling and linking programs if necessary;

when said first process is activated in the user computers at a predetermined time, said first process sends the current configuration information of the object software to said second process in the vendor computer via the network to inquire about the latest configuration, receives the answer from said second process, updates the object software according to the update instruction information in the answer, and prepares for compiling and linking of programs if necessary to realize an automatic update of the object software; and

when a user instructs the user computers to activate said first process, said first process sends the current configuration information of the object software to said second process in the vendor computer via the network to inquire about the latest configuration, receives the answer from said second process, updates the object software according to the update instruction information in the answer, and prepares for compiling and linking of programs if necessary.

10. A software distribution/maintenance system for managing distribution of object software via a network by users who use one or more sets of the object software to be distributed, managed, and maintained and by software vendors who supply the users with the object software, comprising:

first process means for managing plural sets of the object software in each user computer;

vendor software libraries in at least one vendor computer; second process means for providing services relating to said vendor software libraries in the at least one vendor computer; and

a network for connecting each user computer and the at least one vendor computer,

said first process means sending via the network an object software distribution/maintenance request message according to an instruction of the users or an instruction of a user-written program to said second process means of the at least one vendor computer containing desired object software, receiving an answer message from the second process means, and distributing or maintaining the object software; and

said second process means receiving the message from said first process means, referring to said vendor software libraries managed by the software vendors according to the message and settings made by the software vendors, generating an answer message in response to the object software distribution/

maintenance request message, and sending the answer message to said first process means of a message sender.

11. The software distribution/maintenance system according to claim 10, wherein

when any object software managed by said first process means in the user computer abnormally terminates during its operation, said first process means detects and analyzes an abnormal condition, and then automatically sends a fault report to said second process means of the vendors over the network; and

said second process means of the vendors receives the fault report and informs the vendors of the fault report.

12. The software distribution/maintenance system according to claim 10 or claim 11, wherein when each version of the object software is provided for a number of the users by the vendors, a single computer user can be a user of a version of one piece of software and also a vendor of a version of another piece of software, and furthermore, a single computer can be provided with one or more first process means and one or more second process means.

13. The software distribution/maintenance system according to claim 10, claim 11, or claim 12, wherein

each user computer can be designed to store user management data for each of the users or each user group, and the data can be set for each object software; and said first process means refers to the user management data to manage one or more sets of the object software.

14. The software distribution/maintenance system according to claim 13, wherein said user management data includes at least one of user identification information of each user or each user group, software management information of each object software used by each user, software configuration information, and message process method specification information.

15. The software distribution/maintenance system according to claim 14, wherein in specifying a method of processing a message according to the user management data, the users can specify a process method as a procedure for each function performed by said first process means.

16. The software distribution/maintenance system according to claim 15, wherein

in specifying a method of processing a message according to the user management data, a vendor confirmation procedure can be used so that the vendor can be conformed to protect user software.

17. The software distribution/maintenance system according to claim 10, claim 11, or claim 12, wherein when said second process means of the at least one vendor computer offers services using said vendor software libraries, said second process means can refer to vendor management data stored for each vendor software library.

18. The software distribution/maintenance system according to claim 17, wherein said vendor management data includes at least one of vendor identification information, software management information, software configuration information, message process method specification information, customer information, and fault history information.

19. The software distribution/maintenance system according to claim 18, wherein when a message process method is specified using the vendor management data, the software vendors can specify a procedure for each function performed by said second process means.

20. The software distribution/maintenance system according to claim 19, wherein a user confirmation procedure can

be selected when a message process method is specified using the vendor management data.

21. The software distribution/maintenance systems according to claim 10 or claim 11, wherein

Each of the users can simultaneously obtain and use a plurality of versions of each object software.

22. A software distribution/maintenance method for managing distribution of object software via a network by users who use one or more sets of object software to be distributed, managed, and maintained and by software vendors who supply the users with the object software using, in user computers, the object software to be used by the users and a first process for managing one or more sets of object software; in each vendor computer, vendor software libraries and a second process for providing services relating to the vendor software libraries; and the network connecting each user computer to at least one vendor computer, comprising:

activating by a sending user the first process through a command input by the sending user in one of the user computers or through a command from a user program;

sending by the sending user from the first process in the one of the user computers via the network, an object software distribution/maintenance request message to the second process of the at least one vendor computer from which the object software is provided;

receiving by the second process in the at least one vendor computer, from which the object software is provided, the object software distribution/maintenance request message from the first process;

referring by the second process to the vendor software libraries managed by the software vendors according to the object software distribution/maintenance request message and settings made by the software vendors;

generating an answer message in response to the object software distribution/maintenance request message;

sending the answer message to the first process of the sending user;

receiving by the first process in the one of the user computers of the sending user the answer message from the second process; and

performing processes for distributing or maintaining the object software according to the answer message and settings made by the users.

23. The software distribution maintenance method according to claim 22,

wherein when executing object software managed by the first process in one of the user computers abnormally terminates during operation, the first process detects and analyzes an abnormal condition, and then sends a fault report message to a vendor of the executing object software via the network, and

wherein the second process receives the fault report message via the network, and informs the vendor of the abnormal condition.

24. The software distribution/maintenance method according to claim 22 or claim 23, wherein

each user computer stores user management data for each of the users or each user group, identifying one or more sets of the object software; and

said first process refers to the user management data to manage the one or more sets of the object software.

25. The software distribution/maintenance method according to claim 24, wherein the user management data includes at least user identification information of at least one of the users, software management information of each

object software used by each user in an object software group, software configuration information, and message process method specification information; and

said first process refers to the user management data.

26. The software distribution/maintenance method according to claim 25, wherein in referring to a method of processing a message according to the user management data, the sending user can specify and use a process method as a procedure for each function performed by the first process.

27. The software distribution/maintenance method according to claim 26, wherein in referring to the method of processing the message according to the user management data, a user-defined vendor confirmation procedure is used so that the message received by the first process can be recognized as a correct message from a predetermined vendor.

28. The software distribution/maintenance method according to claim 25, wherein when the sending user activates the first process in one of the user computers by inputting a command, the first process

obtains a user name, a command name, and an object software name to generate an interactive screen displaying messages according to the command,

promotes generation of a message in response to the command from the sending user by referring to the user management data of the sending user and the object software or by invoking a user-defined process procedure,

sends the message to the second process of the computer of the software vendors of the object software, and returns to a standby state after recording transmission of the message.

29. The software distribution/maintenance method according to claim 25 or claim 26, wherein if the first process is activated in the one of the user computers by an input command from a program defined by the sending user or through an abnormal termination of the object software managed by the first process, the first process

obtains a user name, a command name, and an object software name,

promotes generation of a message in response to a command from the sending user by referring to the user management data of the sending user and the object software or by invoking a user-defined process procedure,

sends the message to the second process in the at least one vendor computer, and

returns to a standby state after recording transmission of the message.

30. The software distribution/maintenance method according to claim 25 or claim 26, wherein if the first process is activated in the one of the user computers by a message received from the network, then the first process

obtains from a header of the answer message a destination user name identifying a specified user, a command name, and an object software name,

uses the user management data for the specified user and the object software,

then obtains a sender name from the header of the answer message,

confirms that the message was sent by a correct vendor of the object software,

also confirms that the first process is in an answer-wait state,

refers to the user management data or invokes the user-defined process procedure depending on the command in the answer message,

analyzes or stores message data or stores or installs the software specified by the answer message, returns, in response to a specific message which requires acknowledgment, an acknowledgment or a process state in an acknowledgment message to the second process via the network, and

then returns to a standby state after recording transmission of the acknowledgment message.

31. The software distribution/maintenance method according to any of claim 25 to claim 27, wherein said first process performs following processes depending on a cause activating the first process:

when the sending user activates the first process in the sending user computers by inputting a command, the first process

obtains a user name, a command name, and an object software name to generate an interactive screen displaying message according to the input command, promotes generation of a first message in response to the command from the sending user by referring to the user management data of the sending user and the object software or by invoking a user-defined process procedure,

sends the message to the second process of the at least one vendor computer storing the object software, and returns to a standby state after recording transmission of the object software distribution/maintenance request message;

if the first process is activated in the one of the user computers by an input command from a program defined by the sending user or through an abnormal termination of the object software managed by the first process, the first process

obtains a user name, a command name, and an object software name,

promotes in response to an activated command generation of a second message in response to a command from the sending user by referring to the user management data of the sending user and the object software or by invoking a user-defined process procedure,

sends the second message to the second process of the at least one vendor computer storing the object software, and

returns to a standby state after recording transmission of the second message; and

if the first process is activated in the one of the user computers by a received message from the network, then the first process

obtains from a header of the received message a destination user name, a command name, and an object software name,

uses the user management data for a specified user and the object software,

then obtains a sender name from the header of the received message,

confirms that the received message was sent by a correct vendor of the object software,

also confirms that the first process is in an answer-wait state,

refers to the user management data or invokes the user-defined process procedure depending on the command name in the received message,

analyzes or stores message data or stores or installs the software specified by the received message, returns, if the received message requires acknowledgment, an acknowledgment message with an acknowledgment or a process state to the second process via the network, and

then returns to a standby state after recording a transmission of the acknowledgment message.

32. The software distribution/maintenance method according to claim 31, wherein each time the first process is activated, a new process is generated to allow plural processes to be performed in parallel on processes other than for a single set of object software for a single user.

33. The software distribution/maintenance method according to claim 22 or claim 23, wherein when the second process in the at least one vendor computer offers services using the vendor software libraries, vendor management data, stored for each vendor software library, is referred to.

34. The software distribution/maintenance method according to claim 33, wherein when the second process refers to the vendor management data, reference is made corresponding to the vendor software library to at least vendor identification information, software management information, software configuration information, message process method specification information, customer information, update history information, and fault history information.

35. The software distribution/maintenance method according to claim 34, wherein when a message process method is referred to using the vendor management data, a procedure each function performed by the second process is invoked.

36. The software distribution/maintenance method according to claim 35, wherein when the message process method is referred to using the vendor management data, a user confirmation procedure can be invoked to confirm an identification of the sending user of the object software distribution/maintenance request message, determine a qualification of the sending user, and return an appropriate answer to the sending user.

37. The software distribution/maintenance method according to claim 36, wherein in the user confirmation procedure as a part of the vendor management data, the qualification of the sending user can be determined based on commercial relations including at least one of contracts relating to the object software, payments relating to the object software, and payments relating to new services for the object software.

38. The software distribution/maintenance method according to claim 36, wherein in the user confirmation procedure depending on the vendor management data, the object software is provided and distributed only to qualified users by referring to identification information including at least one of organizations, titles, and personal names thereof.

39. The software distribution/maintenance method according to claim 36, wherein

in the user confirmation procedure as a part of the vendor management data, the user can be identified by a password.

40. The software distribution/maintenance method according to claim 22 or claim 23, wherein the one of the user computers and the at least one vendor computer are connected to the network only when sending and receiving messages between each other or a relaying device in the network.

41. The software distribution/maintenance method according to claim 22, or claim 23 wherein the object

software distribution/maintenance request message transmitted via the network has a destination and a source of the message, and a title column specifying a name of a software distribution/maintenance system, a name of a software distribution/maintenance process type, and a name of the object software.

42. The software distribution/maintenance method according to claim 22, claim 23, or any of claims 33 to 39, wherein the second process in the at least one vendor computer is normally in a standby state, and upon being activated by receiving the object software distribution/maintenance request message from the first process via the network

obtains a command name and an object software name of specified software from a header of the object software distribution/maintenance request message,
 assigns processes to be performed,
 obtains a name of the sending user and user identification information from the object software distribution/maintenance request message,
 compares the name of the sending user with vendor management data to determine a user access qualification relating to the command name,
 analyzes the object software distribution/maintenance request message if the sending user qualifies,
 provides summary information of the specified software,
 provides new services for the specified software,
 supplies updated software and instructions to correctly update the specified software, retrieves information of other messages,
 generates the answer message,
 records a summary of the answer message to the sending user,
 sends the answer message to the first process of the sending user via the network, and
 returns to the standby state.

43. The software distribution/maintenance method according to claim 42, wherein each time the second process is activated, a new process is generated to allow a plurality of processes to be concurrently performed on processes other than for a single set of object software for a single user.

44. The software distribution/maintenance method according to claim 22, or claim 23 wherein when the users obtain summary information of the object software not yet used by the users, following processes are performed:

in the one of the user computers,
 the sending user
 enters a summary inquiry command,
 activates the first process in the one of the user computers,
 sets a name of specified object software, a name of a specified vendor, a network address of the specified vendor, and an answer information storage directory, and
 generates a summary information inquiry message;
 the first process sends the summary information inquiry message to the second process of the specified vendor;
 in the at least one vendor computer,
 the second process is activated when the summary information inquiry message is received via the network;
 the second process
 refers to vendor management data about the specified object software cited by the summary information

inquiry message to confirm an identification of the sending user and determine access qualification of the sending user;

invokes, if the sending user is qualified, a vendor-defined procedure to provide summary information in a summary information message; and
 sends the summary information message to the first process of the sending user via the network.

45. The software distribution/maintenance method according to claim 22 or claim 23, wherein when the users newly obtain the object software not yet used by the users, following processes are performed:

in the one of the user computers,
 the sending user
 enters a new installation command,
 activates the first process in the one of the user computers,
 sets user identification information, software information of specified object software, vendor identification information, and user installation process information, and
 generates a new installation request message; and
 the first process sends the new installation request message to the second process managing the specified object software;
 in the at least one vendor computer,
 the second process is activated when the new installation request message is received via the network,
 the second process
 refers to vendor management data about the specified object software to confirm identification of the sending user and determine access qualification of the sending user;
 invokes, if the sending user is qualified, a vendor-defined procedure for new installation to determine a version to be provided,
 obtains a set of modules of the version to be provided,
 organizes instruction information of the new installation in the one of the user computers, and
 generates a new installation message as the answer message,
 sends the new installation message as the answer message to the first process of the sending user via the network;
 in the user computer,
 the first process is activated when the new installation message is received via the network,
 the first process
 obtains a name of a destination user and a name of specified object software from a header of the new installation message,
 refers to corresponding user management data,
 confirms that the first process is currently in a new installation standby state and that a correct vendor sent the new installation message,
 invokes a user-defined new installation process procedure,
 refers to installation information in the new installation message,
 stores the specified object software to be newly installed in the one of the user computers,
 installs the set of modules if the new installation is acceptable,
 records the new installation message and a process result, and
 returns to the standby state.

105

46. The software distribution/maintenance method according to claim 22, or claim 23, wherein when the users add or switch to a new service of the object software already used by the users, following processes are performed:

- in the one of the user computers,
 - the sending user
 - enters a new service request command,
 - activates the first process in the one of the user computers,
 - sets user identification information, software information of specified object software, vendor identification information, and user installation process information, and
 - generates a new service request message;
 - the first process sends the new service request message to the second process of the at least one vendor computer;
- in the at least one vendor computer,
 - the second process is activated when the new service request message is received via the network;
 - the second process
 - refers to vendor management data about the specified object software cited by the new service request to confirm identification of the sending user and determine access qualification of the sending users,
 - invokes, if the sending user is qualified, a vendor-defined procedure for a supply of the new service to determine a version to be provided,
 - obtains a set of modules of the version to be provided,
 - organizes instruction information of installation in the one of the user computers,
 - specifies a process to be performed on an old service, generates a new installation message as an answer, and
 - sends the new installation message as the answer to the first process of the sending user via the network;
- in the user computer,
 - the first process is activated when the new installation message is received via the network;
 - the first process
 - obtains a name of a destination user and the name of the specified object software from a header of the new installation message,
 - refers to corresponding user management data, confirms that the third process is currently in a new installation standby state and a correct vendor sent the new installation message,
 - invokes a user-defined process procedure,
 - refers to installation information in the new installation message,
 - stores specified software supplied as the new service in the one of the user computers,
 - stores the new service of the specified software if the new service is acceptable,
 - records the new installation message and a process result, and
 - returns to the standby state.

47. The software distribution/maintenance method according to claim 22 or claim 23, wherein when the users inquire of the vendors about an update state of installed object software already used by the users and request updated object software, following processes are performed:

- in the one of the user computers,
 - the sending user
 - enters an update inquiry command,
 - activates the first process in the one of the user computers,
 - sets user identification information, software information of the installed object software, vendor identification information, software management information, and user installation process information, and
 - generates an update inquiry message;
 - the first process sends the update inquiry message to the second process of a specified vendor identified by the vendor identification information;
- in the at least one vendor computer,
 - the second process is activated when the update inquiry message is received via the network;
 - the second process
 - refers to vendor management data about the installed object software cited by the update inquiry message to confirm identification of the sending user and determine access qualification of the sending user,
 - invokes, if the sending user is qualified, a vendor-defined update inquiry answering procedure to determine a version type to be supplied and a latest version and compares the latest version with a current version of the sending user;
 - if the current version of the sending user matches the latest version, then the second process generates a non-update message, and sends the non-update message to the first process of the sending user;
 - if the undated object software with the version type of the current version of the installed object software exists, then the second process generates module delete/add instruction information about modules to be added in an update instruction message as an answer, and sends the answer to the first process of the sending user via the network;
 - if the current version should be switched to a newly served version of the installed object software, then the second process generates the non-update message and an information message about the newly served version, and sends the non-update message and the information message to the first process of the sending user via the network;
- in the one of the user computers,
 - the first process is activated when a received message is the update instruction message or the non-update message received via the network;
 - the first process
 - obtains a name of a destination user and a name of specified object software from a header of the received message,
 - refers to corresponding user management data, and confirms that the first process is currently in a standby state after inquiry and that a correct vendor sent the received message;
 - if the non-update message is received, the first process records the non-update message and returns to the standby state;
 - if the update instruction message is received, the first process invokes a user-defined update reception process procedure,
 - refers to the module delete/add instruction information in the received message,

106

enters an update inquiry command,

activates the first process in the one of the user computers,

sets user identification information, software information of the installed object software, vendor identification information, software management information, and user installation process information, and

generates an update inquiry message;

the first process sends the update inquiry message to the second process of a specified vendor identified by the vendor identification information;

in the at least one vendor computer,

the second process is activated when the update inquiry message is received via the network;

the second process

refers to vendor management data about the installed object software cited by the update inquiry message to confirm identification of the sending user and determine access qualification of the sending user,

invokes, if the sending user is qualified, a vendor-defined update inquiry answering procedure to determine a version type to be supplied and a latest version and compares the latest version with a current version of the sending user;

if the current version of the sending user matches the latest version, then the second process generates a non-update message, and sends the non-update message to the first process of the sending user;

if the undated object software with the version type of the current version of the installed object software exists, then the second process generates module delete/add instruction information about modules to be added in an update instruction message as an answer, and sends the answer to the first process of the sending user via the network;

if the current version should be switched to a newly served version of the installed object software, then the second process generates the non-update message and an information message about the newly served version, and sends the non-update message and the information message to the first process of the sending user via the network;

in the one of the user computers,

the first process is activated when a received message is the update instruction message or the non-update message received via the network;

the first process

obtains a name of a destination user and a name of specified object software from a header of the received message,

refers to corresponding user management data, and confirms that the first process is currently in a standby state after inquiry and that a correct vendor sent the received message;

if the non-update message is received, the first process records the non-update message and returns to the standby state;

if the update instruction message is received, the first process invokes a user-defined update reception process procedure,

refers to the module delete/add instruction information in the received message,

107

stores in the one of the user computers the specified object software specified by the update instruction message,
 installs the specified object software if the specified object software is acceptable, and
 records the received message and a process result, and returns to the standby state.

48. The software distribution/maintenance method according to claim 22 or claim 23, wherein when the users inquire about an update state of installed object software already used by the users and request updated object software, following processes are performed:

in the one of the user computers,

the first process

is activated according to an update inquiry command instruction of a user program,
 invokes a user-defined automatically-editing update inquiry procedure,

refers to user management data, and
 generates an update inquiry message;

the first process sends the update inquiry message to the second process of a specified vendor of the installed object software;

in the at least one user computer,

the second process is activated when the update inquiry message is received via the network;
 the second process

refers to vendor management data about the installed object software cited by the update inquiry message to confirm identification of the sending user and determine access qualification of the sending user, and

invokes, if the sending user is qualified, a vendor-defined update inquiry answering procedure to determine a version type to be supplied and a latest version and compare the latest version with the current version of the sending user;

if the current version of the sending user matches the latest version, then the second process generates a non-update message, and sends the non-update message to the first process of the sending user;

if the installed object software of the version type of the current version of the sending user has been updated, then the second process

generates module delete/add instruction information and information about modules to be added in an update instruction message as an answer, and sends the updated instruction message to the first process of the sending user via the network;

if the current version should be switched to a newly served version of the installed object software, then the second process

generates a non-update message and an information message about the newly served version, and sends the non-update message to the first process of the sending user via the network;

in the user computer,

the first process is activated when the update instruction message or the non-update message is received from the second process via the network;
 the first process

obtains a name of a destination user and a name of the object software from a header of a received message,
 refers to corresponding user management data, and confirms that the first process is currently in a standby state after inquiry and that a correct vendor sent the received message;

108

if the non-update message is received, the first process records the received message and returns to the standby state;

if the update instruction message is received, the first process

invokes a user-defined update reception process procedure,

refers to the module delete/add instruction information in the received message,

stores in the one of the user computers the modules to be added specified by the update instruction message,

installs the modules to be added if acceptable, records the received message and a process result, and

returns to the standby state.

49. The software distribution/maintenance method according to any of claim 45 to claim 48, wherein if the first process receives a message from a sending vendor, and stores or installs in the one of the user computers new software, newly served software, or updated software received from the sending vendor, then the first process monitors a result of processes and sends to the second process via the network a process result confirmation message informing whether the processes have terminated normally or abnormally.

50. A software distribution/maintenance system having a plurality of user computers and a vendor computer to manage and automatically update object software provided for the plurality of user computers from the vendor computer, comprising:

network means for connecting the user computers to the vendor computer; and

process means in each of the user computers, for sending current configuration information of the object software stored in the user computers to the vendor computer via said network means to inquire about a latest configuration, for receiving an answer from the vendor computer, and for updating the object software stored in the user computers according to an instruction in a received answer.

51. The software distribution/maintenance system according to the claim 50, wherein said process means in the user computers automatically informs via said network means the vendor computer of an abnormal termination and state if an operation of the object software abnormally terminates in the user computers.

52. The software distribution/maintenance system according to claim 51, wherein said process means automatically informs the vendor computer of the abnormal termination, an instruction causing the abnormal termination, a reason for the abnormal termination, a series of instructions which invoked the instruction causing the abnormal termination, and an environment of software and hardware used when the abnormal termination is detected.

53. The software distribution/maintenance system according to claim 51, wherein said process means automatically informs via said network means the vendor computer of the abnormal termination by general-purpose electronic mail.

54. A software distribution/maintenance system having a plurality of user computers and a vendor computer to manage and automatically update object software provided for the plurality of user computers from the vendor computer, comprising:

network means for connecting the user computers to the vendor computers; and

process means in the vendor computer for receiving an inquiry from the user computer, for generating update

instruction information for the object software to match a configuration of the object software in the user computer with the configuration of an updated version in an object software library stored in the vendor computer, and for returning via said network means to the user computer the update instruction information and the updated version of the object software.

55. A software distribution/maintenance method for managing and automatically updating a set of object software stored in user computers from a vendor computer via a network, comprising:

sending by a process in the user computers current configuration information of the object software to the vendor computer via the network to inquire about a latest configuration;

receiving by the process an answer with update instruction information from the vendor computer; and

updating the object software according to the update instruction information about the object software to match a configuration of the object software with the latest configuration stored in an object software library in the vendor computer, including preparing for compiling and linking of programs if necessary.

56. The software distribution/maintenance method according to claim 55, wherein when the object software is activated in the user computers, the process in the user computers

immediately sends the current configuration information of the object software to the vendor computer via the network to inquire about the latest configuration, receives the answer from the vendor computer, updates the object software according to the update instruction information in the answer, and prepares for compiling and linking of programs if necessary.

57. The software distribution/maintenance method according to claim 55, wherein when the process is activated in the user computers at a predetermined time, the process sends the current configuration information of the object software to the vendor computer via the network to inquire about the latest configuration, receives the answer from the vendor computer, updates the object software according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary to realize an automatic update of the object software.

58. The software distribution/maintenance method according to claim 55, wherein when a user instructs the user computers to activate the process, the process

sends the current configuration information of the object software to the vendor computer via the network to inquire about the latest configuration,

receives the answer from the vendor computer, updates the object software according to the update instruction information in the answer, and

prepares for the compiling and linking of programs if necessary.

59. The software distribution/maintenance method according to claim 55, wherein following processes are interchangeably performed:

when the object software is activated in the user computers, the process

immediately sends the current configuration information of the object software to the vendor computer

via the network to inquire about the latest configuration,

receives the answer from the vendor computer, updates the object software according to the update instruction information in the answer, and prepares for compiling and linking of programs if necessary;

when the process is activated in the user computers at a predetermined time, the process

sends the current configuration information of the object software to the vendor computer via the network to inquire about the latest configuration, receives the answer from the vendor computer, updates the object software according to the update instruction information in the answer, and

prepares for the compiling and linking of programs if necessary to realize an automatic update of the object software; and

when a user instructs the user computers to activate the process, the process

sends the current configuration information of the object software to the vendor computer via the network to inquire about the latest configuration, receives the answer from the vendor computer, updates the object software according to the update instruction information in the answer, and prepares for the compiling and linking of programs if necessary.

60. A software distribution/maintenance method for managing and automatically updating a set of object software stored in user computers from a vendor computer via a network, comprising:

receiving by a process stored in the vendor computer an inquiry from an inquiring user computer about current configuration information of the object software, in the user computer;

generating update instruction information about the object software to update the current configuration of the object software in the user computer with a latest configuration version in an object software library in the vendor computer; and

returning the update instruction information and an updated version of the object software to the inquiring user computer via the network.

61. A software distribution/maintenance system for managing distribution of object software via a network accessed by users of user computers who use sets of the object software distributed, managed, and maintained by software vendors who supply the users with the object software from at least one vendor computer, comprising:

a network for connecting each user computer and the at least one vendor computer; and

process means in each user computer for executing the object software to be used by the users, for managing the sets of object software in each user computer, for sending via the network an object software distribution/maintenance request message according to an instruction of the users or an instruction of a user-written program to the at least one vendor computer which has a software library containing the object software, for receiving an answer message from the at least one vendor computer, and for distributing or maintaining the object software.

62. The software distribution/maintenance system according to claim 61, wherein when any object software managed by said process means in each user computer abnormally

terminates during operation, said process means detects and analyzes an abnormal condition, and then automatically sends a fault report to the at least one vendor computer via said network.

63. The software distribution/maintenance system according to claim 61, wherein

each user computer can store user management data for each of the users or each user group, and for each set of the object software; and

said process means refers to the user management data to manage one or more sets of the object software.

64. The software distribution/maintenance system according to claim 63, wherein said user management data includes user identification information of each user or each user group, software management information of each set of the object software used by each user, software configuration information, and message process method specification information.

65. The software distribution/maintenance system according to claim 64, wherein in specifying a method of processing a message according to the user management data, the users can specify a process method as a procedure for each function performed by said process means.

66. The software distribution/maintenance system according to claim 65, wherein in specifying a method of processing a message according to the user management data, a vendor confirmation procedure can be used so that the vendor can be confirmed to protect user software.

67. A software distribution/maintenance system for managing distribution of object software via a communications network accessed by users of user computers use sets of the object software distributed, managed, and maintained by software vendors who supply the users with the object software from at least one vendor computer, comprising:

network means for connecting each user computer and the at least one vendor computer;

vendor software libraries; and

process means for providing services relating to said vendor software libraries, for receiving an object software distribution/maintenance request message from an inquiring user computer via said network means, for referring to one of said vendor software libraries, managed by a vendors according to contents of the object software distribution/maintenance request message and settings of the vendor, for generating an answer message in response to the object software distribution/maintenance request message, and for sending the answer message to the inquiring user computer via said network means.

68. The software distribution/maintenance system according to claim 67, wherein when any object software abnormally terminates during operation in one of the user computers, said process means sends a fault report received from the one of the user computers via said network means to the vendor of the object software which abnormally terminated.

69. The software distribution/maintenance system according to claim 67, wherein when said process means of the at least one vendor computer offers services using said vendor software libraries, said process means refers to vendor management data stored for each vendor software library.

70. The software distribution/maintenance system according to claim 69, wherein

said vendor management data can be designed to comprise vendor identification information, software management information, software configuration

information, message process method specification information, customer information, and fault history information.

71. The software distribution/maintenance system according to claim 70, wherein when a message process method is specified using the vendor management data, the software vendors can specify the message process method as a procedure for each function performed by said process means.

72. The software distribution/maintenance system according to claim 70, wherein

a user confirmation procedure can be selected when a message process method is specified using the vendor management data.

73. A software distribution/maintenance method for managing distribution of object software via a network accessed by users of user computers who use sets of the object software distributed, managed, and maintained by software vendors who supply the users with the object software from at least one vendor computer using, in each user computer, the object software to be used by the users and a process managing the sets of the object software, and the network connecting each user computer and the at least one vendor computer, comprising:

activating by the users the process through a command input by the users or through a command from a user-written program;

sending by the process via the network an object software distribution/maintenance request message to the at least one vendor computer from which the object software is provided;

receiving by the process the answer message from the at least one vendor computer; and

performing processes for at least one of distributing and maintaining the object software according to the answer message and settings made by the users.

74. A software distribution/maintenance method for managing distribution of object software via a network accessed by users of user computers who use sets of the object software distributed, managed, and maintained by software vendors who supply the users with the object software using, in each vendor computer, vendor software libraries and a vendor process providing services relating to the vendor software libraries, the network connecting each user computer and each vendor computer, said software distribution/maintenance method comprising:

receiving by the vendor process an object software distribution/maintenance request message from one of the user computers via the network;

referring by the vendor process to the vendor software libraries managed by one of the vendors according to the object software distribution/maintenance request message and settings made by the one of the vendors; generating an answer message in response to the object software distribution/maintenance request message; and

sending the answer message to the one of the user computers which issued the object software distribution/maintenance request message.

75. A software distribution and maintenance system using a network in which a plurality of users using a number of types of object software to be distributed, managed, and maintained, and a plurality of software vendors supplying the object software manage the distribution and maintenance of the object software over a computer network, comprising: one or more first process means CP installed in each of user computers, that manage a plurality of object soft-

ware groups to be used by one or more users individually for every object software and for every user; one or more second process means installed in one or more software vendor computers, that each providing services to a Plurality of vendor software libraries for each of the software libraries; and
 a network that connects the first process means installed in the user computer to the second process means, based on standardized communications protocols, said first process means having a capability to perform distribution or maintenance of the object software by sending a message a request of distribution or maintenance of the object software for one piece of object software over the network, according to instructions given by one or more of the users or a user defined program, receiving an answer message from the second process means SP, and processing based on contents of the answer message and settings made by the users, and

said second process means having a capability to receive the message from each first process means to reference one or more of the software libraries managed by one or more of the vendors depending on the contents of the message and settings made by the vendors for the object software specified with the message, to generate the answer message to answer the request of distribution or maintenance of the object software, and to send the answer message to said first process means that generated the message.

76. The software distribution and maintenance system using a network according to claim 75, used by the users taking advantage of one or more pieces of the object software included in the object software groups, and the vendors supplying one or more pieces of the object software in the object software groups, wherein one or more computers each comprise both said one or more first process programs and said one or more second process means.

77. The software distribution and maintenance system using a network according to claim 75, wherein the message, sent and received over said network between said first process means installed in one of the user computers, and said second process means installed in one of the vendor computers, consists of a header part of the message, an attribute part as fixed data, an instructing part as a specific request and instruction data, and a module part as corresponding software.

78. The software distribution and maintenance system using a network according to claim 75, wherein the attribute part forming a part of the message is configured as hierarchical structure data of a pair of a keyword and a value, facilitates referencing and updating management data for user set for every object software and for every user, and management data for vendor set for every object software library and for every vendor.

79. User computers used in a software distribution and maintenance system using a network in which a plurality of users using a plurality of types of object software to be distributed, managed, and maintained, and a plurality of software vendors supplying the object software manage the distribution and maintenance of the object software over said network, comprising:

one or more process means that manage a plurality of object software groups to be used by the users individually for every object software and for every user using the object software; and

storage means for storing management data for user set for every user for every object software and user

process procedures which are process procedures for said process means to distribute and maintain the object software, set for every object software and for every user, said process means calling said user process procedures to reference the management data for user, send a message to request distribution and maintenance of the object software to the vendor of the object software over said network according to standardized protocols in the system, and perform distribution and maintenance of the target software.

80. The user computers in the software distribution and maintenance system using a network according to claim 79, wherein the management data for user includes user identification information corresponding to each user, vendor identification information corresponding to each object software vendor, software identification information corresponding to each object software, and environment information.

81. The user computers in the software distribution and maintenance system using a network according to claim 79, wherein each user process procedure comprises:

message-editing procedures that edit a message to request distribution and maintenance sent from said user process procedure to the vendor of the object software;

a vendor verification procedure that verifies a vendor of the object software; and

reception process procedures that process an answer message from the vendor of the object software.

82. The user computers in the software distribution and maintenance system using a network according to claim 79, each comprises said management data for user, said user process procedures, and said first process means for each of the plurality of versions of the object software so that every user can utilize the plurality of versions.

83. The user computers in the software distribution and maintenance system using a network according to claim 79, each comprises data for specifying a plurality of versions for said management data for user, and both said user process procedures and said first process means for the plurality of versions so that every user can utilize a plurality of versions of an identical piece of the object software at the same time.

84. A vendor computer in a software distribution and maintenance system using a network in which a plurality of users using a plurality of types of object software to be distributed, managed, and maintained, and a plurality of software vendors supplying the object software manage the object software distribution for distributing the object software over said network, comprising:

at least one process means for providing services of each of the software libraries and each of the vendor software libraries; and

storage means for storing management data for vendor set for every vendor and for every software library, and vendor process procedures which are process procedures for said process means to distribute and maintain the object software on the user computers, set for every vendor and for every software library, said process means performing a corresponding process for a message of the object software from the user, set for every vendor and for every object software library.

85. The vendor computer in the software distribution and maintenance system using a network according to claim 84, wherein said management data for each vendor includes user identification information concerning all the users of the object software, vendor identification information concerning the vendor of the object software library, software

information concerning the object software library, and environment information.

86. The vendor computer in the software distribution and maintenance system using a network according to claim 84, wherein said vendor process procedures include a user qualification verification procedure for the object software, an unqualified user process procedure, reception and answer process procedures for analyzing a message received from the user, and generating an answer message.

87. The vendor computer in the software distribution and maintenance system using a network according to claim 84, wherein said object software library includes a plurality of versions of the object software, and said process means corresponding to the object software library gives services to said object software library using management data for vendor and vendor process procedures that correspond to a plurality of versions.

88. A software distribution and maintenance method in which a plurality of users using a plurality of types of object software to be distributed, managed, and maintained, and a plurality of software vendors supplying the object software, manage distribution and maintenance of the object software, said method utilizing at least one first process in each of a plurality of user computers, that manage object software groups to be used by the users individually for every object software and for every user using the object software, at least one second process in vendor computers, that provide services to vendor software libraries for each of the vendor software libraries, and a network that connects the user computers to the vendor computers standardized communication protocols, said method comprising:

invoking a first process with a command entered by one of the users on one of the user computers or with a command invoked from a user-defined program;

sending a request message from the first process to request distribution or maintenance of at least one piece of the object software to a second process managing at least one of the vendor software libraries of the object software;

receiving the request message by the second process in one of the vendor computers;

generating an answer message to answer the request message by referencing at least one of the vendor software libraries managed by at least one of the software vendors for the at least one piece of the object software specified by the request message;

returning the answer message to the first process;

receiving the answer message by the first process; and processing the answer message depending on contents of the answer message and settings made by the one of the users for the distribution or maintenance of the at least one piece of the object software.

89. The software distribution and maintenance method using a network according to claim 88, wherein the first process means installed in the user computers, which is invoked with a command entered by the user on the user computers, or a command in a user-defined program, calls a message-editing procedure which is one of user process procedures for distributing and maintaining the object software, edits a message to request distribution and maintenance based on the standardized protocols within the system while referencing the management data for user set for message and generating an answer message, references the object software library while referencing and updating said management data for vendor, and generates an answer message for distribution and maintenance of the object

software in response to the received message to request distribution and maintenance, and returns the answer message to the first process means.

90. The software distribution and maintenance method using a network according to claim 88, wherein the first process installed in the user computers,

receives the answer message returned from the second process in response to the request message sent by the first process to request distribution or maintenance of the object software,

verifies that the answer message is returned from a legal vendor,

calls a reception process procedure which is one of user process procedures set for the one of the users and for the object software,

performs distribution and maintenance of the object software according to contents of the answer message, and prepares for execution of the object software according to a determination made by the first process while referencing and updating management data for user set for the one of the users and for the object software.

91. The software distribution and maintenance method using a network according to claim 88, wherein the first process installed in the one of the user computers

receives the answer message returned by the second process in response to the request message sent by the first process to request distribution and maintenance of the object software,

performs distribution and maintenance of the object software according to the answer message, and

then automatically returns a response message to the second process indicating whether the distribution and maintenance was properly executed on the one of the user computers.

92. The software distribution and maintenance method using a network according to claim 88, wherein said second process installed in the vendor computers

receives the request message from the first process to request distribution and maintenance of the object software belonging to the vendor software libraries managed by the second process,

verifies that the request message to request distribution and maintenance is sent from a legal user by referencing user identification information included in vendor management data set for the software vendors and for the vendor software libraries,

calls a reception/answer process procedure for analyzing the request message and generating the answer message,

references the at least one of the vendor software libraries while referencing and updating the vendor management data,

generates the answer message for distribution and maintenance of the object software in response to the request message to request distribution and maintenance, and returns the answer message to the first process.

93. The software distribution and maintenance method using a network according to claim 88, wherein a message, which is sent and received over the network between the first process means (CP) installed in one of the user computers and the second process means installed in one of the vendor computers, consists of a header part of the message, an attribute part as fixed data, an instructing part, a specific request and instruction data, an instructing part as instruction data, and a module part as corresponding software.

117

94. A method for purchasing new software in a software distribution and maintenance system using a network, in which a plurality of users using a plurality of types of object software to be distributed, managed, and maintained, and a number of software vendors supplying the object software manage the distribution and maintenance of the object software over a network, comprising:

one of the users sending a message to request summary information of the object software from first process means in the user computers to second process means over said network by use of any of the first process means which simply has the capability to request summary information and of receiving an answer message to the request of summary information;

the second process means providing basic components of the first process means and user process procedures sufficient for a new purchase of the object software as

118

part of the answer message in response to the summary inquiry to the user computers over the network;

the user sending a message to request a new purchase to the second process means over the network using the basic components of the first process means and the user process procedures;

the second process means providing the whole set of said first process means and user process procedures appropriate for the distribution and maintenance of the object software to the user's first process means over the network;

the user computers thereafter distributing and maintaining the object software using the first process means and the user process procedures.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 5,835,911
DATED : November 10, 1998
INVENTOR(S) : Nakagawa et al.

PAGE 1 of 4

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title Page, [73], "Kanagawa" should be --Kawasaki--.

Col. 4, line 24, "method" should be --method.--.

Col. 9, line 54, BEGIN A NEW PARAGRAPH with "The second..."

Col. 10, line 17, "stared" should be --stored--;
line 49, delete "is".

Col. 11, line 54, BEGIN A NEW PARAGRAPH with "The first...";
line 62, "1a" should be --3a--.

Col. 13, line 2, BEGIN A NEW PARAGRAPH with "The user...";
line 17, "U2," should be --U2,....--.

Col. 18, line 37, "user" should be --user.--.

Col. 22, line 38, BEGIN A NEW PARAGRAPH with "Furthermore..";
line 55, BEGIN A NEW PARAGRAPH with "If the..."

Col. 24, line 16, "S11" should be -- SL1--;
line 59, "SL2," should be --SL2,....--.

Col. 27, line 17, "De" should be --be--;
line 40, "intra-of fice" should be --intra-office--.

Col. 28, line 32, "by" should be --by an--;
line 61, "stores;" should be --stores--.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 5,835,911

DATED : November 10, 1998

INVENTOR(S) : Nakagawa et al.

PAGE 2 of 4

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

- Col. 32, line 25, "13" should be --B--;
line 28, after "information" insert --is--.
- Col. 36, line 3, "software" should be --software--;
line 49, "follow ing" should be --following--.
- Col. 37, line 6, "Asoftware" should be --A software--.
- Col. 38, line 16, "Fig. 6" should be --Fig. 8--.
- Col. 42, [Table 2], "ANSER" should be --ANSWER--.
- Col. 43, line 2, "process" should be --process--;
line 16, "a" should be --□--;
line 17, "A" should be --Δ--;
line 37, "Δ" should be --□--.
- Col. 45, line 15, "0" should be --□--;
line 36, "Δ" should be --□--.
- Col. 46, line 27, "of" should be --of user--;
line 62, "□" should be --0--.
- Col. 55, line 50, " 1" " should be --1'--.
- Col. 56, line 35, BEGIN A NEW PARAGRAPH with "In step.."
- Col. 67, line 5, "auser" should be --a user--.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 5,835,911
DATED : November 10, 1998
INVENTOR(S) : Nakagawa et al.

PAGE 3 of 4

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

- Col. 72, line 53, "S1" should be --SL--.
- Col. 74, line 5, "With-these" should be --With these--;
line 52, User 1" should be --User 1@--;
line 53, BEGIN A NEW PARAGRAPH with "Subject:..."
- Col. 75, line 11, "itself" should be --itself.--;
line 29, "is" should be --is,--;
line 32, "mb.2" should be --Mb.2--.
- Col. 76, line 34, "data" should be --data)--.
- Col. 78, line 39, delete entire line.
- Col. 80, line 9, "Me.2" should be --Mb.2--.
- Col. 82, line 63, "Object" should be --object--.
- Col. 84, line 27, BEGIN A NEW PARAGRAPH with "The software..."
- Col. 87, line 45, delete ",",.
- Col. 91, line 59, delete "Step S78:";
line 60, before "The" insert new line with --Step S78--.
- Col. 92, line 37, "(C?)" should be --(CP)--;
line 43, "383" should be --S83--.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 5,835,911
DATED : November 10, 1998
INVENTOR(S) : Nakagawa et al.

PAGE 4 of 4

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

- Col. 94, line 48, after "software." continue with line 50
(NO NEW PARAGRAPH).
- Col. 98, line 67, after "wherein" START A NEW PARAGRAPH with
"a user...".
- Col. 99, line 5, "Each" should be --each--;
line 65, "after "wherein" START A NEW PARAGRAPH with
"the user...".
- Col. 101, line 5, after "message," START A NEW LINE with
"returns,...".
- Col. 111, line 44, "vendors" should be --vendor,--.
- Col. 112, line 66, delete "CP".
- Col. 113, line 5, "Plurality" should be --plurality--.
- Col. 115, line 30, after "computers" (second occurrence) insert
--and uses--.

Signed and Sealed this
Twentieth Day of July, 1999

Attest:



Q. TODD DICKINSON

Attesting Officer

Acting Commissioner of Patents and Trademarks

EXHIBIT D

United States Patent [19]

[11] Patent Number: 5,797,016

Chen et al.

[45] Date of Patent: Aug. 18, 1998

- [54] REGENERATION AGENT FOR BACK-UP SOFTWARE
- [75] Inventors: Chia-Hwang Chen, Plainview; Miguel Long, Brooklyn; William Hsieh, Syosset, all of N.Y.
- [73] Assignees: Cheyenne Software Inc.; Cheyenne Software International Sales Corp., both of Roslyn Heights, N.Y.
- [21] Appl. No.: 743,459
- [22] Filed: Oct. 29, 1996
- [51] Int. Cl.⁶ G06F 11/14; G06F 13/00
- [52] U.S. Cl. 395/712; 395/653
- [58] Field of Search 395/712, 652

5,226,157	7/1993	Nakano et al.	711/162
5,237,661	8/1993	Kawamura et al.	395/872
5,247,626	9/1993	Firoozmand	395/200.42
5,247,670	9/1993	Matsunaga	395/200.33
5,274,815	12/1993	Trissel et al.	395/567
5,276,860	1/1994	Fortier et al.	395/182.04
5,297,195	3/1994	Thorne et al.	379/93.23
5,313,637	5/1994	Rose	395/186
5,317,691	5/1994	Traeger	395/681
5,321,816	6/1994	Rogan et al.	711/163
5,324,035	6/1994	Morris et al.	463/42
5,339,430	8/1994	Laundin et al.	395/685
5,365,577	11/1994	Davis et al.	379/93.17
5,452,433	9/1995	Nihart et al.	711/162
5,469,573	11/1995	McGill, III et al.	395/653
5,495,610	2/1996	Shing et al.	395/200.51
5,544,320	8/1996	Konrad	395/200.09
5,634,052	5/1997	Morris	395/601
5,664,186	9/1997	Bennett et al.	711/162

[56] References Cited

U.S. PATENT DOCUMENTS

4,177,510	12/1979	Appel et al.	711/163
4,725,977	2/1988	Izumi et al.	711/115
4,751,648	6/1988	Sears, III et al.	707/102
4,856,787	8/1989	Idkis	273/237
5,056,000	10/1991	Chang	395/800.21
5,086,502	2/1992	Malcolm	395/182.06
5,101,479	3/1992	Baker et al.	395/285
5,109,384	4/1992	Tseung	395/182.02
5,131,081	7/1992	MacKenna et al.	395/842
5,133,065	7/1992	Cheffetz et al.	395/181
5,138,712	8/1992	Corbin	395/186
5,144,551	9/1992	Cepulis	711/163
5,163,131	11/1992	Row et al.	395/200.32
5,170,466	12/1992	Rogan et al.	707/530
5,185,693	2/1993	Lofus et al.	395/182.11
5,187,750	2/1993	Behera	707/7
5,204,954	4/1993	Hammer et al.	395/842
5,212,772	3/1993	Masters	395/182.18
5,218,695	6/1993	Noveck et al.	707/205
5,222,122	6/1993	Hamilton et al.	379/32

Primary Examiner—Emanuel Todd Voeltz

Assistant Examiner—John Q. Chavis

Attorney, Agent, or Firm—Kenyon & Kenyon

[57]

ABSTRACT

A system for updating an agent used in a backup software program. The backup software program operates on a network having, for example, a server and a number of workstations. A backup engine executes on the server. An agent executes on each of the workstations to be backed up. The backup engine transmits an updated agent to each workstation, transmits an executable regeneration module to each workstation and transmits an execute command to the agent at each workstation. The agent at each workstation stores the updated agent and the executable regeneration module and causes the execution of the executable regeneration module. The executable regeneration module deletes or renames the agent, and also renames the updated agent to the name of the agent and thereafter enables operation of the updated agent as the agent.

14 Claims, 4 Drawing Sheets



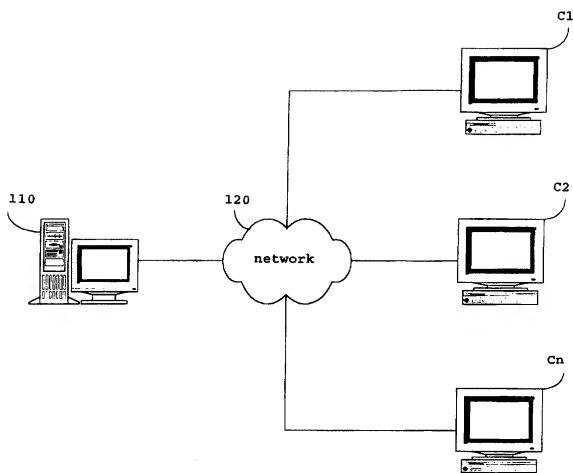


Fig. 1

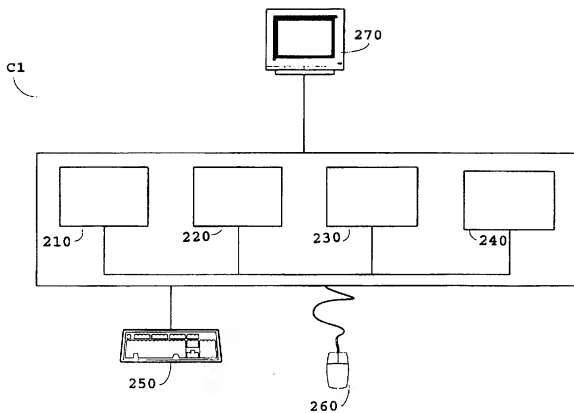


Fig. 2

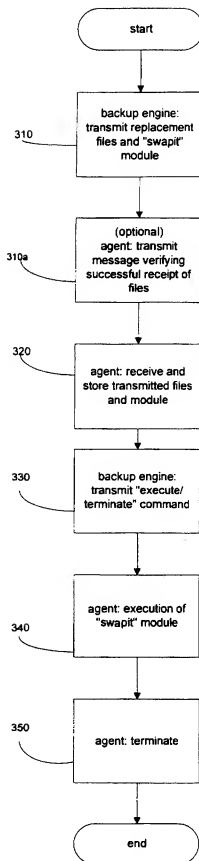


Fig. 3

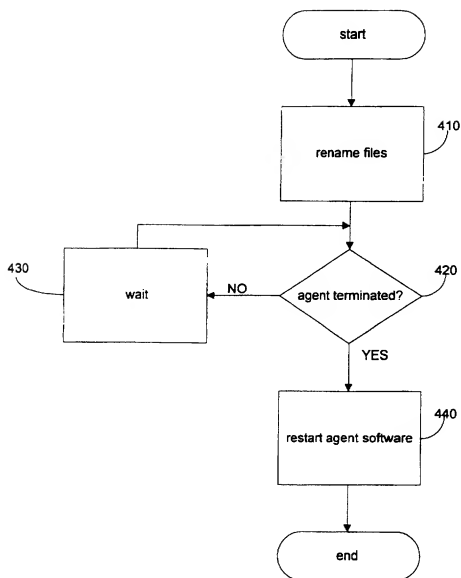


Fig. 4

1

REGENERATION AGENT FOR BACK-UP SOFTWARE

FIELD OF INVENTION

The present invention is directed to agent modules used in back-up software systems, and more particularly, to a system and method for updating agent modules on a remote workstation.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND OF THE INVENTION

Modern computer networks are often organized according to a client/server architecture. A client/server architecture is an arrangement used on local area networks ("LANs") that treats both the server and the individual workstations as intelligent, programmable devices. Typically, a LAN comprises a number of "front-end" client computers and a "back-end" server. The client component, typically a personal computer, offers the user its full range of power and features for running programs. Usually, the client computer (called herein a workstation) has its own processing capability and a hard disk drive or other local storage device. The server, which can be a personal computer, a minicomputer or a mainframe, enhances the client component by providing services such as data management, information sharing and security. Servers provide services to workstations including, for example, back-up services.

Often, a user of the LAN (typically a network manager) wishes to back-up data stored on the hard drive(s) of some or all of the computers on the LAN, including the workstations. In the back-up process, the files stored on workstations and servers of the LAN are down-loaded onto a central storage device, such as a tape on a tape drive. Thus, for example, if a file is damaged on a workstation, the network manager can retrieve the back-up copy of the lost data from the central back-up storage device.

A typical back-up program has a number of components. A backup engine, running centrally on the server, can control the writing of data to the backup storage device (for a backup job), control the reading of data from the backup storage device (for a restore job), manage a task queue of jobs, and control communications with the client computers. An administrator console, which typically runs on a client computer, assists the network manager manage the backup of workstations. For example, the administrator console will perform tasks such as job submission, viewing log files, database management, scheduling and the like.

Back-up programs often include agents. An agent is a small piece of software that is stored on and processed by each workstation to perform slave tasks for the server. Thus, the agent runs on each computer to be backed up. The agent is not a complete computer program, but rather, a piece of software to support the server in completing a task defined for the workstation on which the agent is running. In backup programs, one job of the agent is to move data from the workstation to the server and receive data from the server and store it at the workstation.

2

Using agents to assist in the backup and restore process has a number of advantages. Because there is an agent running on each workstation (and each workstation, of course, has its own processor), processing can take place concurrently on each workstation. Communication overhead is reduced and network security is enhanced, as the agent can, for example, determine the content of a workstation's storage device and make decisions as to files to be backed up, without the need to provide the server with a list of files stored on the workstation.

A disadvantage of current agent technology in backup programs results from the fact that agents are located at each workstation. Thus, if the network administrator wishes to update the backup program to a newer release of the program, the network administrator will often need to locate and update each agent on each workstation. Since there may be many workstations located in many physical locations, this could be a difficult and time consuming task.

In particular, in a distributed application, it is difficult to synchronize the updating of agents on the plurality of workstations.

The need to update agents is likely to become more common. For example, agents for various backup programs are often bundled with operating system programs for workstations. A user will then purchase the backup program, comprising the backup engine and administrator console. The backup engine will communicate with each workstation, as each workstation has an agent preinstalled with the operating system. However, when the backup program is updated to a new release, the backup engine and administrator console can easily be replaced/updated, but it is difficult and time consuming to replace/update each agent.

There is also a need to apply updates and cause the updated agents to be available for use by the backup engine at the same time.

SUMMARY OF THE INVENTION

The present invention is directed to a method and system for updating or replacing agents stored and executing on a remote computer, in particular, a workstation in a client/server network. In the representative embodiment, the agents are utilized in a backup software program and perform slave tasks for a backup engine running on a central server.

Each agent is a small piece of software that is stored and executes on each workstation to support the backup engine in completing a task defined for that workstation. Thus, in summary, each agent is a module that resides on a workstation that is being backed up or restored and performs, e.g., file processing on that workstation.

In the representative embodiment, the backup program that utilizes the present invention has three major components. The backup engine, located on the server, controls the writing of data to the backup storage device (for a backup job), controls the reading of data from the backup storage device (for a restore job), manages a task queue of jobs, and controls communications with the client computers. An administrator console, which runs on a client computer, enables management of the backup and restore operations. As stated above, an agent runs on each workstation to be backed up.

For ease of explanation, when used herein, the term "backup" includes "restore", the term "update" includes "replace" and "modify", and the term "LAN" includes a wide area network ("WAN") and an enterprise wide network.

3

The updating of an agent occurs according to the representative embodiment of the present invention as follows. Assume that each agent is named A1. The backup engine electronically transmits, across the network to each workstation, a replacement agent file (A2) that comprises the executable code for the agent to replace A1. The backup engine also electronically transmits to each workstation an executable regeneration module, for example, named "Swapit". A1 and Swapit are stored by A1 at the workstation.

Optionally, each agent A1 transmits a message to the backup engine verifying successful receipt of A2 and Swapit.

Thereafter, the backup engine transmits to each workstation (i.e., to A1 on each workstation) an "execute/terminate" command. Each agent A1 then loads and causes to be executed by its workstation the Swapit module, and then each agent A1 terminates its own operation.

The Swapit module, executing on each workstation, renames A1 to preserve a copy of A1. The Swapit module then renames A2 to a new name appropriate for execution, for example, to A1. The Swapit module causes A1 to execute or to be available for execution, for example, upon receipt of a command from the backup engine.

If desirable, the present invention can be used to update a subset of the agents on a LAN.

Each workstation may have more than one backup agent. The present invention is scalable and can easily accommodate updating of more than one agent stored on a workstation.

If the update operation is not successful, the present invention has the capability to allow a graceful backing out of the process and restoration of the old agent (e.g., A1 in the example above).

The present invention can coordinate the timing of the update. For example, for a large LAN, the transmission of the new agent (e.g., A2) can occur over a number of evenings, and execution of the Swapit module can take place on all workstations simultaneously.

The backup agent of the present invention has capabilities to enable the efficient backup of the storage devices on all workstations on the network. When used in conjunction with the update process, network communication overheads are reduced.

It will be noted, for example, that the backup engine need only communicate with the workstations on two occasions (when transferring the new agent and the Swapit module and when issuing the "execute/terminate" instruction). All processing is performed "automatically" either by the "old" agent or the Swapit module, both executing on the workstation, rather than in response to a series of commands from the server.

Compression and decompression techniques also can be used to reduce transmission times and intermediate storage costs.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other features of the present invention will be more readily apparent from the following detailed description of exemplary embodiments taken in conjunction with the attached drawings wherein:

FIG. 1 is a system diagram of a network architecture utilized by the invention.

FIG. 2 is a block diagram of an illustrative hardware configuration of a client system.

4

FIG. 3 is a flowchart of an exemplary update process of a representative embodiment of the present invention.

FIG. 4 is a flowchart of the operation of an exemplary update module.

DETAILED DESCRIPTION

Referring now to the drawings and initially FIG. 1, there is illustrated a diagram of a representative network used in conjunction with the present invention. A server 110 is coupled to client computers C1-Cn through a network 120. The network 120 may be any type of network that supports computer-to-computer communication such as a local area network (LAN), a wide area network (WAN), or a public switched network (PSN) supporting any range of protocols such as TCP/IP, Ethernet, X.25, etc. The server 110 executes a backup engine program. The server 110 is coupled to a backup storage device (not shown).

Each of the client computers C1-Cn is a workstation having an operating system program, such as, for example, the Windows 95 or Windows NT operating system. Each client computer C1-Cn stores and can execute (when instructed to do so by the backup engine) an agent. Each agent performs certain tasks for the backup engine at each of the client computers C1-Cn. The agent is configured, for example, to open and/or connect to a socket and "listen" for commands directed to it from the backup engine on the server 110. For example, the backup engine may request that each agent "push" selected files to the server 110 so that the server 110 (under control of the backup engine) can copy the files to the backup storage device.

The following is a more detailed illustrative description of some of the functions that can be performed by the agent of the present invention. A backup job may specify particular files that are to be backed up or criteria for files to be backed up (e.g., all files created by Bob, all WordPerfect files, etc.). The backup job is created at the administrator console, which runs on a client computer (e.g., C2), and is executed by the backup engine on the server 110. If a specified file is to be backed up, the backup engine sends a request to the agent running on the client computer where the file is located, and the agent checks to see if the file is available for backup, and sends a copy of the file to the backup engine. If criteria is used to identify the files to be backed up, the backup engine provides the criteria to the agents on the appropriate client computers. Each agent will then traverse the directory structure of the storage devices. If a file is located by the agent that matches the criteria, a copy of the file is sent by the agent to the backup engine. Alternatively, a circular buffer can be used—the agentA adds file names that match the criteria to the buffer and agentB performs the read/write operation to send that file to the backup engine. Thus, it will be seen that the agent supports, as a slave, the server in completing a task defined for that client computer.

FIG. 2 illustrates in further detail the hardware configuration of the client computer (e.g., C1) of FIG. 1. In the representative embodiment, the client computer C1 comprises a central processing unit 210 for executing computer programs (including the agent according to the present invention) and managing and controlling the operation of the client computer C1. Storage device 220, such as a floppy disk drive, is coupled to the central processing unit 210. Storage device 230, coupled to the central processing unit 210, also provides a means for storing computer programs and data. Storage device 230 is preferably a hard disk having a high storage capacity. A dynamic memory device 240, such as a RAM, is coupled to the central processing unit 210. The

5

client computer C1 includes typical input/output devices, such as, for example, a keyboard 250, a mouse 260 and a monitor 270. Each of the remaining client computers C2-Cn may be similarly configured. The server 110 may also be similarly configured but may further include connections to a plurality of high capacity storage media.

According to the present invention, agent on each client computer C1-Cn is updated from the server 110. Thus, the "new" agent is received from a storage device on the server. Alternatively, the "new" agent may be originally stored on, and therefore received from, any other computer connected to the network 120 and communicating with the client computers C1-Cn.

FIG. 3 is a flowchart illustrating a representative update process according to the present invention.

In accordance with an exemplary embodiment of the invention, the backup engine first transmits (or causes to be transmitted) to each client computer C1-Cn (or to a subset thereof) files containing the updated agent (step 310). Also transmitted is a "Swapit" module that is described in detail in connection with the flow chart of FIG. 4 (step 310).

The "old" agent (executing on each client computer C1-Cn) receives and stores the files transmitted by the workstation (step 310). The received files are stored locally at the client computer C1-Cn (e.g., in storage device 230) and may have a file extension of "new" to indicate a new file. Thus, for example, "agent.new" may represent the new executable agent (to be later renamed). The "old" agent may then optionally transmit messages back to the backup engine verifying the files were successfully received and stored (step 310a).

Next, the backup engine transmits an "execute/terminate" command to the appropriate client computers C1-Cn (step 330). The execute/terminate command may include a parameter identifying a procedure for the client computer C1-Cn to execute (e.g., in this case "Swapit.") In response thereto, each agent loads and causes to be executed the program identified in the parameter, e.g., Swapit (step 340) and then shuts itself down (step 350).

The flowchart of FIG. 4 provides details of an exemplary program flow of the Swapit module of steps 310 and 340 of FIG. 3. The Swapit module executes on each client computer C1-Cn. The Swapit module may execute concurrently on each client computer C1-Cn. Upon commencement of execution, the Swapit module renames the "old" agent to a new name so as to preserve a copy of that version of the agent (step 410). For example, "agent.exe" may be renamed to "agent.old." (Alternatively, the Swapit module may simply delete the old agent.) The Swapit module then renames the files associated with the updated agent to a new name appropriate for execution (step 410). E.g., "agent.new" is renamed to "agent.exe."

Once updated, the new agent can remain ready to act as the slave of the backup engine. Alternatively, it may be desirable to cause the new agent to execute upon the end of the update process. Thus, the following additional steps take place at the client computer. Subsequent to renaming all appropriate files, the Swapit module determines whether the old agent is still executing (step 420). This may be accomplished, for example, by invoking a Windows NT command which returns the "handle" of an executing module. If a handle is returned when the handle for the agent module is requested, the agent is still executing. In that case, the Swapit module waits for a predetermined length of time (step 430) and then checks again for a "handle" (step 420). If a handle is not returned, the old agent has terminated. The

6

Swapit module then starts the agent software, i.e., the updated agent (step 440).

It will be appreciated that steps 310 to 320 can be performed over a period of time, for example, when network use is low. Once all client computers C1-Cn have received the updated agent and the Swapit module, then the "execute/terminate" command can be sent substantially simultaneously to all client computers C1-Cn where updating is to take place.

If desirable, the new agent and the Swapit module can be transmitted to the client computers C1-Cn in one package or, alternatively, in separate files at separate times.

Optionally, the new agent can delete the Swapit module.

An variation to the above may include the transmission of software patch to the agent rather than a replacement agent. In that case, the Swapit module would i) make a backup copy of the old agent, ii) apply the software patches to the old agent, and iii) make the agent available for execution.

In an alternative embodiment, the Swapit module can be configured to be self executing, for example, at a predetermined time. Thus, there would be no need for step 330. The Swapit module would, at step 350, determine if the agent was executing, and pause operation until a time when the agent was not executing.

It will be appreciated that the principles of the present invention can be applied to the updating of any agent executing on a remote system, not merely backup agents.

The agent and Swapit modules of the present invention can be implemented utilizing a logic circuit or a computer memory comprising encoded computer-readable instructions, such as a computer program. The functionality of the logic circuit or computer memory is described in detail above.

While the present invention has been particularly shown and described with reference to exemplary embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. A method for updating an agent used in a backup software program, the backup software program operating on a network having a server and a plurality of workstations, the backup software program including a backup engine executing on the server and an agent executing on each workstation to be backed up, the method comprising the steps of:

- under the control of the backup engine, transmitting an updated agent to the agent at each workstation;
- under the control of the backup engine, transmitting an executable regeneration module to the agent at each workstation;
- under the control of the agent at each workstation, storing the updated agent and the executable regeneration module at each workstation;
- under the control of the backup engine, transmitting an execute command to the agent at each workstation;
- under the control of the agent at each workstation, causing the execution of the executable regeneration module; terminating the operation of the agent;
- under the control of the executable regeneration module at each workstation, deleting the agent;
- under the control of the executable regeneration module at each workstation, renaming the updated agent to the name of the agent; and

under the control of the executable regeneration module at each workstation, enabling operation of the updated agent as the agent.

2. The method of claim 1 further comprising the step of, under the control of the agent at each workstation, notifying the backup engine of successful receipt of the updated agent and the executable regeneration module.

3. The method of claim 1 further comprising the step of deleting the executable regeneration module.

4. The method of claim 1 wherein the step of transmitting an execute command to the agent at each workstation occurs so as to enable simultaneous updating of each agent.

5. A method for updating an agent used in a backup software program, the backup software program operating on a network having a server and a plurality of workstations, the server coupled to a backup storage device, the method comprising the steps of:

providing a backup engine for execution on the server; providing an agent for execution on each of the plurality of workstations as a slave of the backup engine;

under the control of the backup engine, transmitting an updated agent to each workstation;

under the control of the backup engine, transmitting an executable regeneration module to each workstation; under the control of the agent at each workstation, storing the updated agent and the executable regeneration module at each workstation;

under the control of the backup engine, transmitting an execute command to the agent at each workstation;

under the control of the agent at each workstation, causing the execution of the executable regeneration module; terminating the operation of the agent;

under the control of the executable regeneration module at each workstation, renaming the agent to a recovery name;

under the control of the executable regeneration module at each workstation, renaming the updated agent to the name of the agent; and

under the control of the executable regeneration module at each workstation, enabling operation of the updated agent as the agent.

6. The method of claim 5 further comprising the step of executing the agent at a workstation to assist the backup engine in a backup operation where files stored on the workstation are backed up across the network to the backup storage device.

7. The method of claim 6 further comprising the step of, under the control of the agent at each workstation, notifying the backup engine of successful receipt of the updated agent and the executable regeneration module.

8. The method of claim 6 further comprising the step of deleting the executable regeneration module.

9. The method of claim 6 wherein the step of transmitting an execute command to the agent at each workstation occurs so as to enable simultaneous updating of each agent.

10. The method of claim 5 further comprising the step of restoring the agent by renaming the agent having the recovery name to be that of the agent.

11. A system for updating an agent used in a backup software program, the backup software program operating on a network having a server and a plurality of workstations, the system comprising:

a backup engine executing on the server; and

an agent executing on each of the plurality of workstations to be backed up;

the backup engine transmitting an updated agent to each workstation and transmitting an executable regeneration module to each workstation and transmitting an execute command to the agent at each workstation;

the agent at each workstation storing the updated agent and the executable regeneration module at each workstation and causing the execution of the executable regeneration module;

the executable regeneration module at each workstation, deleting the agent and renaming the updated agent to the name of the agent and thereafter enabling operation of the updated agent as the agent.

12. A system for updating an agent used in a backup software program, the backup software program operating on a network having a server and a plurality of workstations, the server coupled to a backup storage device, the system comprising:

a backup engine for execution on the server, the backup engine including means for transmitting an updated agent to each workstation,

means for transmitting an executable regeneration module to each workstation, and means for transmitting an execute command to the agent at each workstation; and

an agent at each workstation, the agent including means for storing the updated agent and the executable regeneration module at each workstation, and causing the execution of the executable regeneration module;

wherein the executable regeneration module at each workstation renames the agent to a recovery name, renames the updated agent to the name of the agent, and enables operation of the updated agent as the agent.

13. The system of claim 12 further comprising means for executing the agent at a workstation to assist the backup engine in a backup operation where files stored on the workstation are backed up across the network to the backup storage device.

14. The system of claim 12 further comprising means for notifying the backup engine of successful receipt of the updated agent and the executable regeneration module.

* * * * *

EXHIBIT E



US006018654A

United States Patent [19][11] **Patent Number:****6,018,654****Valentine et al.**[45] **Date of Patent:****Jan. 25, 2000**[54] **METHOD AND APPARATUS FOR
DOWNLOADING TONES TO MOBILE
TERMINALS**

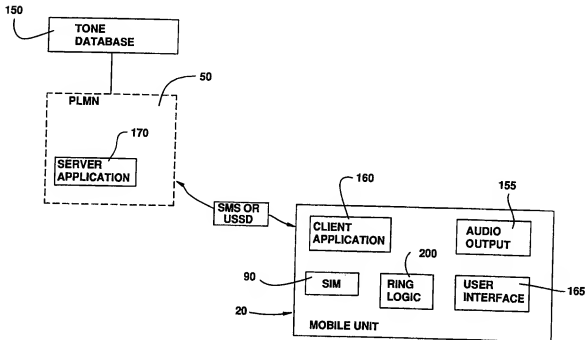
5,481,599 1/1996 MacAllister et al 379/101.01

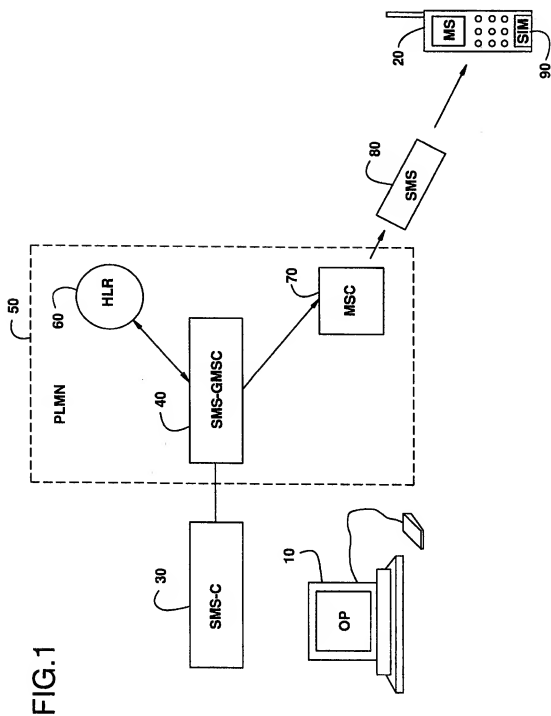
FOREIGN PATENT DOCUMENTS[75] Inventors: **Eric Lee Valentine; Jim Mills**, both of
Plano, Tex.0 562 890 9/1993 European Pat. Off. .
WO 93/26132 12/1993 WIPO .[73] Assignee: **Ericsson Inc.**, Research Triangle Park,
N.C.*Primary Examiner*—Andrew M. Dolinar
Attorney, Agent, or Firm—Jenkins & Gilchrist[21] Appl. No.: **08/739,623**[57] **ABSTRACT**[22] Filed: **Oct. 29, 1996**[51] Int. Cl.⁷ **H04Q 7/22; H04Q 7/32**[52] U.S. Cl. **455/414; 455/419; 455/567**[58] **Field of Search** 455/414, 415,
455/419, 466, 551, 567; 379/101.01, 374,
375

A method and apparatus for downloading tone data from a public land mobile network to a mobile telephone unit is disclosed. A mobile telephone unit includes means enabling the user to request downloading of tone data to the mobile telephone unit from a public land mobile network via a connection-less communications link such as the USSD or SMS. The downloaded tone data is uniquely associated with a selected telephone number within the mobile telephone unit such that a call to the mobile unit involving the telephone number initiates audio play back of the tone data.

[56] **References Cited****U.S. PATENT DOCUMENTS**

5,371,781 12/1994 Ardon 455/445

54 Claims, 4 Drawing Sheets



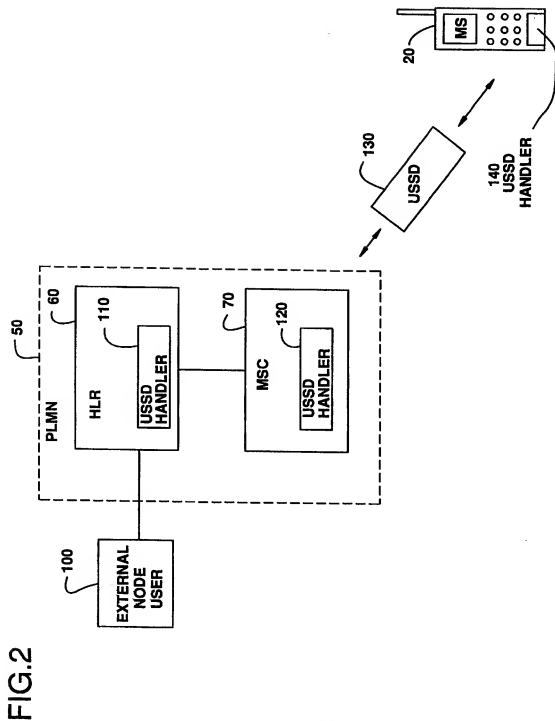


FIG. 3

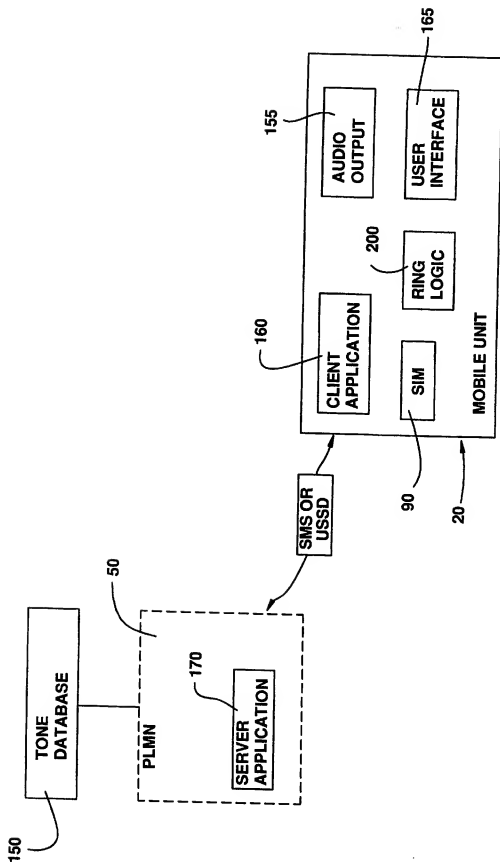
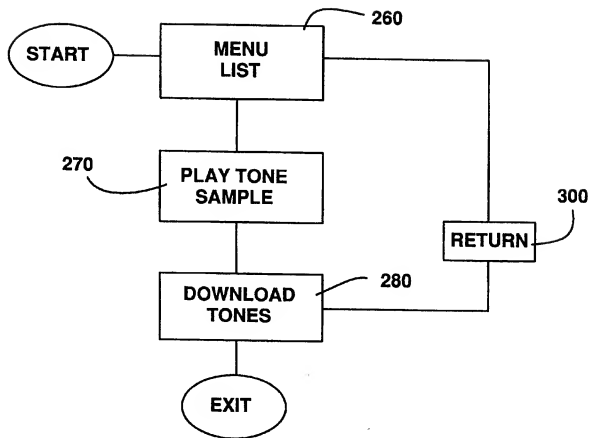


FIG. 4



METHOD AND APPARATUS FOR DOWNLOADING TONES TO MOBILE TERMINALS

BACKGROUND OF THE INVENTION

1. Technical Field of the Invention

The present invention relates to personal communication systems, and more particularly, to the downloading of tone data to a mobile terminal to enable the playing of the tones in association with a particular telephone number.

2. Description of Related Art

The ever expanding list of services available via personal communication services (PCS) systems have provided PCS users with the ability to select a number of services from their mobile telephone unit in addition to the standard telephone communication services. A number of these services require the user to view some type of graphical or alphanumeric display upon the mobile telephone unit. Having to view the display can in some cases be inconvenient, for example, if the user happens to be driving, if the telephone is located in the user's pocket or briefcase, or if the user is involved in activity precluding the use of their hands. Thus, it would be beneficial to enable the user to know who is calling without having to check the calling number display.

In other cases using existing PCS technologies, the user may have more than one telephone number associated with a particular mobile telephone unit, for example, a personal telephone number and a business telephone number. The user can benefit by knowing whether the personal or business number has been called by the use of an indicator that does not require the user to look at the phone. This will enable the user to answer the mobile telephone unit differently based upon whether the business number or personal number was called. Thus, a mobile telephone unit providing the user with the option to select and download new tones to be used for different call scenarios would provide an ease of use and flexibility that would greatly benefit the user.

SUMMARY OF THE INVENTION

The present invention overcomes the foregoing and other problems with a method and apparatus for downloading tone data between a public land mobile network (PLMN) and a mobile unit. A mobile unit includes a client application for requesting the downloading of tone data from a PLMN through a connection-less communications link. Requests from the client application are received by a server application located within the public land mobile network. The server application is normally associated with the mobile switching center (MSC). The server application provides access to a tone data base wherein a user may select a tone for downloading through the mobile unit's user interface.

Once a tone is selected, the tone data associated with the tone is downloaded to the mobile unit via the connection-less user interface. The interface preferably comprises either short message service (SMS) messages or unstructured supplemental services data (USSD) messages which are useful for downloading unstructured user designated data. The downloaded tone data is then uniquely associated with a selected called or calling party telephone number, or group of numbers, such that when a call to the mobile unit involves the selected telephone number, an audio play back of the tone data is initiated.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the method and apparatus of the present invention may be obtained by reference

to the following Detailed Description when taken in conjunction with the accompanying Drawings wherein:

FIG. 1 is a block diagram illustrating the communication of a short message service (SMS) messages between a SMS operator and a mobile station;

FIG. 2 is a block diagram illustrating the communication of an unstructured supplemental services data (USSD) messages between a USSD external node user and a mobile station;

FIG. 3 is a block diagram illustrating the components necessary for downloading of tones between a PLMN and mobile unit; and

FIG. 4 is a block diagram illustrating the manner through which a user interactively downloads tones to a subscriber identity module (SIM) card.

DETAILED DESCRIPTION OF THE DRAWINGS

Telecommunication services are normally performed in a structured way. For example, specific predefined data, formats, and signal names are used to set up a speech connection, to perform handovers, and to authenticate mobile subscriber information when providing telecommunication service to a mobile subscriber. With the introduction of the global system for mobile communications (GSM) and the personal communications systems (PCS), a number of new and advanced supplementary services are being provided to mobile subscribers. Since these supplementary services utilize user specified data, there are no structured ways to communicate this data between the public land mobile network (PLMN) and a mobile station. As a result, a number of unstructured business protocols have been developed for the GSM or PCS environments. As the transmission of tone data between a PLMN and a mobile unit falls under the category of transmitting unstructured user data, the transfer would be controlled by one of these protocols.

Once such protocol is unstructured supplementary service data (USSD) which has been introduced to enable user interaction between PLMN applications and a mobile station in a transparent way through a mobile telecommunication network. The communication is transparent because no review or manipulation of the contents of the message is performed during the transportation period.

One type of user specified information that may be transmitted between a PLMN and a mobile telephone unit is tone data, which then may be associated with called or calling numbers in a manner designated by the user. Reference is now made to FIG. 1, where a block diagram generally illustrates the communication of a short message service (SMS) message between a SMS operator 10 and mobile station 20. The SMS operator 10 sends data to the short message service center (SMS-C) 30 to be transmitted to the mobile station 20. The SMS-C 30 encapsulates the entered data into a packet message, such as signaling system number 7 (SS 7) signals or X.25 protocol packets, and routes the message to a short message service-gateway mobile switch center (SMS-GMSC) 40 within a PLMN 50 serving the mobile station 20. The SMS-GMSC 40 interrogates a home location register (HLR) 60 associated with the mobile unit 20 for routing information (i.e., an identification where the mobile station 20 is currently located) and subsequently routes the message to a mobile switching center (MSC) 70 serving the mobile station's current location. The mobile station 20 is paged and a connection is set up between the mobile station and the PLMN 50.

If the mobile station 20 is already busy, the connection setup is not performed because the network already knows

the mobile station 20 is accessible. If the connection has been successful and thereby the mobile station 20 authenticated, the MSC 70 encapsulates the tone data into an SMS message 80 and delivers the SMS message to the mobile station 20 over one of the control data channels. The control data channel such as a stand alone dedicated control channel (SDCCH) is used instead of a traffic channel (TCH) to allow connection-less data communication. After receiving the SMS message 80 encapsulating the tone data, the mobile station acts merely as a buffer and passes the data to the attached subscriber identity module (SIM) card 90. The SIM card 90 then stores the received data into an internal buffer or memory register. Lastly, if the delivery has been successful, a successful delivery report is sent back from the MS 20 to the serving MSC 70, and subsequently from the serving MSC 70 to the SMS-C 30. Otherwise, a failure report is generated.

With respect to a mobile originated SMS message (MO-SMS) a user at a mobile station 20 can initiate an SMS signal to request downloading of data, such as tone data. The mobile station 20 makes a request to the mobile switching center (MSC) 70 to transmit tone data to the mobile station 20. The MSC 70 encapsulates the request into a packet message, and routes the message to a short message service gateway mobile switch center (SMS-GMSC) 40 within a PLMN 50 serving the mobile station 20. The SMS-GMSC 40 retrieves the requested data and subsequently routes a message to the MSC 70 serving the mobile station's current location. The mobile station 20 is then paged and a connection is set up between the mobile station and the PLMN 50. The MSC 70 encapsulates the tone data into an SMS message 80 and delivers the SMS message to the mobile station 20 over one of the control data channels. The data is then stored within the SIM card 90 as previously described.

FIG. 2 is a block diagram illustrating the communication of a USSD message between a USSD external node user 100 and a mobile station 20. USSD messages are utilized by the mobile telecommunications network to transport user defined data to a mobile station 20 or an application module within a mobile station. Therefore, instead of storing and receiving character data into a SIM card, the received data is either manipulated by the feature application modules within the receiving mobile station to provide special subscriber feature functions, or it is displayed on a display unit for user interactions.

The external node user 100 transmits the USSD message encapsulating the tone data to the HLR 60 within the serving PLMN 50. The HLR 60 is associated within a number of different MSC's within the same PLMN 50. As the mobile station 20 travels from one MSC's area to another, the HLR receives location update signals into record of the mobile station's current location. Whenever a USSD signal is received by the HLR, the HLR ascertains a current location of the mobile station 20. The USSD handler 110 within the HLR 60 thereafter transparently forwards the USSD signal to the appropriate MSC 70 currently serving the mobile station 20. The USSD handler 120 within the serving MSC 70 receives the transmitted message and transports the USSD message 130 to the mobile station 20 over a connection-less communications link. The USSD handler 140 within the mobile station 20 then receives the transmitted USSD message 130, extracts encapsulated tone data, and forwards the extracted data to the appropriate application module.

Referring now to FIG. 3 where a block diagram illustrates the components necessary for downloading tones requested by a subscriber (user). The subscriber requests access to a

tone database 150 containing a variety of predetermined data packages representing a particular tone or group of tones to be played by the audio output 155 of the mobile unit 20 in response to a particular calling or called number. The client application 160 within the mobile unit 20 initiates a request for access to the tone database in response to inputs by the user through the user interface 165. The client application 160 actuates a serving application 170 located with the PLMN. The serving application 170 may be located with the MSC/VLR, the HLR, or some other external node. The serving application 170 connects the user with the database 150 using either the SMS or USSD protocols discussed earlier. The user then selects desired tones in a manner which will be more fully described with respect to FIG. 4.

The tone data associated with the tone selected from the tone database 150 is downloaded to the client application 160 as a digitally coded tone pattern using either the USSD or SMS protocols described previously with respect to FIGS. 1 and 2. The above-described manner of downloading a tone from the tone database 150 is utilized with respect to menu driven options solely using SMS or USSD messages. optionally, an audio menu may be provided to the user such that an actual connection is generated between the mobile station 20 and the tone database 150. In this case, an audio version of the tones would be played for the user and the client application 160 would record the tone and convert it to a digital format for storage in the SIM card 90.

In the case of a transmission using a SMS message, the serving MSC 70 receives the transmitted tone signal from the SMS-GSMC 40 and then transmits an SMS message encapsulating the tone data to the mobile unit station 20 over a connection-less communication link such as SDCCH. The client application 160 within the mobile unit 20 acts as a buffer for the SMS message and passes the tone data from the message to the SIM card 90. The user may then, through client application 160, associate the tone within the SIM card 90 with a particular calling or called telephone number.

If a USSD message is used for downloading, the tone data is routed to the mobile station 20 by a USSD handler 120 within the serving MSC 70 as a USSD message encapsulating the tone data over a connection-less communication link such as SDCCH. USSD handler 140 within the mobile station receives the transmitted USSD message and forwards the message to the client application 160 for extraction of the tone data. The extracted tone data is then stored within the SIM card 90. Through the client application 160, the user may then uniquely associate the tone with a particular calling or called telephone number.

Once the tone data is downloaded into the SIM card 90 of the mobile unit 20 and associated with a particular telephone number, the receipt of an incoming call actuates ring logic 200 within the mobile terminal 20. The ring logic 200 checks for the presence of tone pattern associated with the number called or the number of the party calling. If such an association is found, the tone data is played by the audio output 155 to provide an audio indicator to the user of who is calling or which of the user's numbers is being called.

Referring now to FIG. 4, there is illustrated the procedure by which a user may download a particular tone pattern from the tone database 150. Once the mobile unit 20 has interconnected with the tone database 150, the mobile unit user is presented at step 260 with a variety of menus enabling the selection of tones by the user. The menus may break the tones down in a variety of manners such as alphabetically, by music type, by novelty items, etc. Once a particular tone is selected, the user may play a sample tone at step 270 to

preview what the tone sounds like. When a desired tone or tone pattern is found, the user may instruct the application 160 to download the tones at step 280. Otherwise, a user may return to previous menus at step 300.

Although an embodiment of the method and apparatus of the present invention has been illustrated in the accompanying Drawings and described in the foregoing Detailed Description, it will be understood that the invention is not limited to the embodiment disclosed, but is capable of numerous rearrangements, modifications and substitutions without departing from the spirit of the invention as set forth and defined by the following claims.

What is claimed is:

1. A mobile station, comprising:

a receiver for receiving tone data over a wireless connection-less communications link from a public land mobile network;

a subscriber identity module card for storing the tone data; and

means for requesting downloading of the tone data to the subscriber identity module card from the public land mobile network over the wireless connection-less communications link and for associating a telephone number with the tone data such that call connections involving the telephone number initiate audio playback of the tone data.

2. The mobile station according to claim 1, wherein the wireless connection-less communications link comprises short message service messages.

3. The mobile station according to claim 1, wherein the wireless connection-less communications link comprises unstructured supplementary service data messages.

4. The mobile station according to claim 1, wherein the telephone number comprises a telephone number of a calling party.

5. The mobile station according to claim 1, wherein the telephone number comprises a telephone number of a called party.

6. The mobile station according to claim 1, further comprising control logic for determining if an incoming call involves the telephone number and initiating audio playback of the tone data for incoming calls involving the telephone number.

7. A system for downloading tone data to a mobile station, comprising:

a public land mobile network serving said mobile station, said public land mobile network including a first application module responsive to a request from the mobile station for downloading said tone data to the mobile station via a wireless connection-less communications link; and

said mobile station comprising:

a subscriber identity module card for storing said tone data;

a receiver for receiving said tone data from the public land mobile network over the wireless connection-less communications link; and

means for requesting downloading of the tone data to the subscriber identity module card from the public land mobile network over the wireless connection-less communications link and for associating a telephone number with the tone data such that call connections involving the telephone number initiate audio playback of the tone data.

8. The system according to claim 7, wherein the wireless connection-less communications link comprises short message service messages.

9. The system according to claim 7, wherein the wireless connection-less communications link comprises unstructured supplementary service data messages.

10. The system according to claim 7, wherein the telephone number comprises a telephone number of a calling party.

11. The system according to claim 7, wherein the telephone number comprises a telephone number of a called party.

12. The system according to claim 7, further comprising control logic for determining if an incoming call involves the telephone number and initiating audio playback of the tone data for incoming calls involving the telephone number.

13. A method for downloading tone data, comprising the steps of:

accessing a public land mobile network using a wireless communications link from a mobile unit;

requesting access to tone data located within the public land mobile network from a client application within the mobile unit;

downloading the requested tone data to a subscriber identity module card of the mobile unit through said wireless connection-less communications link; and associating the downloaded tone data with a selected telephone number.

14. The method according to claim 13, further comprising the step of playing an audio rendition of the tone data in response to receipt by the mobile unit of a call involving the selected telephone number.

15. The mobile station according to claim 13, wherein the wireless communications link comprises short message service messages.

16. The mobile station claim 13, wherein the wireless communications link comprises unstructured supplementary service data messages.

17. The mobile station according to claim 13, wherein the telephone number comprises a telephone number of a calling party.

18. The mobile station according to claim 13, wherein the telephone number comprises a telephone number of a called party.

19. The mobile station according to claim 13, wherein said step of requesting access provides audio playback of the tone data.

20. The mobile station according to claim 13, wherein said step of requesting access provides a text selection of the tone data.

21. A mobile station, comprising:

a receiver for receiving tone data over a connection-less communications link from a public land mobile network;

a register for storing the tone data; and

means for requesting downloading of the tone data to the register from the public land mobile network over the connection-less communications link and for associating a telephone number with the tone data such that call connections involving the telephone number initiate audio playback of the tone data,

wherein said connection-less communications link comprises a link selected from the group consisting of short message service messages and unstructured supplementary service data messages.

22. The mobile station according to claim 21, wherein the register comprises a subscriber identity module card attachable to the mobile station.

23. The mobile station according to claim 21, wherein the telephone number comprises a telephone number of a calling party.

24. The mobile station according to claim 21, wherein the telephone number comprises a telephone number of a called party.

25. The mobile station according to claim 21, further comprising control logic for determining if an incoming call involves the telephone number and initiating audio playback of the tone data for incoming calls involving the telephone number.

26. A mobile station, comprising:

a receiver for receiving tone data over a connection-less communications link from a public land mobile network;

a register for storing the tone data; and

means for requesting downloading of the tone data to the register from the public land mobile network over the connection-less communications link and for associating a telephone number with the tone data such that call connections involving the telephone number initiate audio playback of the tone data, wherein said telephone number comprises a telephone number of a called party.

27. The mobile station according to claim 26, wherein the register comprises a subscriber identity module card attachable to the mobile station.

28. The mobile station according to claim 26, wherein the connection-less communications link comprises short message service messages.

29. The mobile station according to claim 26, wherein the connection-less communications link comprises unstructured supplementary service data messages.

30. The mobile station according to claim 26, wherein the telephone number comprises a telephone number of a calling party.

31. The mobile station according to claim 26, further comprising control logic for determining if an incoming call involves the telephone number and initiating audio playback of the tone data for incoming calls involving the telephone number.

32. A system for downloading tone data to a mobile station, comprising:

a public land mobile network serving said mobile station, said public land mobile network including a first application module responsive to a request from the mobile station for downloading said tone data to the mobile station via a connection-less communications link; and

said mobile station comprising:

a register for storing said tone data;

a receiver for receiving said tone data from the public land mobile network over the connection-less communications link; and

means for requesting downloading of the tone data to the register from the public land mobile network over the connection-less communications link and for associating a telephone number with the tone data such that call connections involving the telephone number initiate audio playback of the tone data, wherein said connection-less communications link comprises a link selected from the group consisting of short message service messages and unstructured supplementary service data messages.

33. The system according to claim 32, wherein the telephone number comprises a telephone number of a calling party.

34. The system according to claim 32, wherein the telephone number comprises a telephone number of a called party.

35. The system according to claim 32, further comprising control logic for determining if an incoming call involves

the telephone number and initiating audio playback of the tone data for incoming calls involving the telephone number.

36. A system for downloading tone data to a mobile station, comprising:

a public land mobile network serving said mobile station, said public land mobile network including a first application module responsive to a request from the mobile station for downloading said tone data to the mobile station via a connection-less communications link; and said mobile station comprising:

a register for storing said tone data;

a receiver for receiving said tone data from the public land mobile network over the connection-less communications link; and

means for requesting downloading of the tone data to the register from the public land mobile network over the connection-less communications link and for associating a telephone number with the tone data such that call connections involving the telephone number initiate audio playback of the tone data, wherein said telephone number comprises a telephone number of a called party.

37. The system according to claim 36, wherein the register comprises a subscriber identity module card attachable to the mobile station.

38. The system according to claim 36, wherein said connection-less communications link comprises short message service messages.

39. The system according to claim 36, wherein said connection-less communications link comprises unstructured supplementary service data messages.

40. The system according to claim 36, wherein the telephone number comprises a telephone number of a calling party.

41. The system according to claim 36, further comprising control logic for determining if an incoming call involves the telephone number and initiating audio playback of the tone data for incoming calls involving the telephone number.

42. A method for downloading tone data, comprising the steps of:

accessing a public land mobile network using a communications link from a mobile unit, said communications link comprising a link selected from the group consisting of short message service messages and unstructured supplementary service data messages;

requesting access to tone data located within the public land mobile network from a client application within the mobile unit;

downloading the requested tone data to the mobile unit through said communications link; and

associating the downloaded tone data with a selected telephone number.

43. The method according to claim 42, further comprising the step of playing an audio rendition of the tone data in response to receipt by the mobile unit of a call involving the selected telephone number.

44. The method according to claim 42, wherein the telephone number comprises a telephone number of a calling party.

45. The method according to claim 42, wherein the telephone number comprises a telephone number of a called party.

46. The method according to claim 42, wherein said step of requesting access provides audio playback of the tone data.

47. The method according to claim 42, wherein said step of requesting access provides a text selection of the tone data.

9

48. A method for downloading tone data, comprising the steps of:

accessing a public land mobile network using a communications link from a mobile unit;

requesting access to tone data located within the public land mobile network from a client application within the mobile unit;

downloading the requested tone data to the mobile unit through said communications link; and

associating the downloaded tone data with a selected telephone number, said selected telephone number comprising a telephone number of a called party.

49. The method according to claim 48, further comprising the step of playing an audio rendition of the tone data in response to receipt by the mobile unit of a call involving the selected telephone number.

10

50. The method according to claim 48, wherein said communications link comprises short message service messages.

51. The method according to claim 48, wherein said communications link comprises unstructured supplementary service data messages.

52. The method according to claim 48, wherein the telephone number comprises a telephone number of a calling party.

53. The method according to claim 48, wherein said step of requesting access provides audio playback of the tone data.

54. The method according to claim 48, wherein said step of requesting access provides a text selection of the tone data.

* * * * *

EXHIBIT F



US005742668A

United States Patent [19][11] **Patent Number:** 5,742,668

Pepe et al.

[45] **Date of Patent:** Apr. 21, 1998

- [54] **ELECTRONIC MASSAGING NETWORK**
- [75] **Inventors:** David Matthew Pepe, Middletown;
 Lisa B. Blitz, Manalapan; James
 Joseph Brockman, Perrineville;
 William Cruz, Eatontown; Dwight
 Omar Hakim, Matawan, all of N.J.;
 Michael Kramer, Bronx County, N.Y.;
 Dawn Diane Petr, Basking Ridge, N.J.;
 Josefa Ramarosan, Frehold, N.J.;
 Gerardo Ramirez, Bridgewater, N.J.;
 Yang-Wei Wang, Howell, N.J.; Robert
 G. White, Morristown, N.J.

5,168,271	12/1992	Hoff	379/60 X
5,311,576	5/1994	Brunson et al.	379/89
5,325,419	6/1994	Connolly et al.	379/60
5,329,579	7/1994	Brunson	379/88
5,351,235	9/1994	Lahinen	370/58.1
5,384,831	1/1995	Cresswell	379/67
5,418,835	5/1995	Frohman et al.	379/57
5,420,911	5/1995	Dahlin et al.	379/59
5,452,289	9/1995	Sharma et al.	370/32.1
5,479,411	12/1995	Klein	370/110.1
5,604,788	2/1997	Tett	379/58

- [73] **Assignee:** Bell Communications Research, Inc.
 Morristown, N.J.

Primary Examiner—Dwayne Bost
Assistant Examiner—Scott Richardson
Attorney, Agent, or Firm—Loria B. Yeaton; Joseph
 Giordano

- [21] **Appl. No.:** 466,623

- [22] **Filed:** Jun. 6, 1995

Related U.S. Application Data

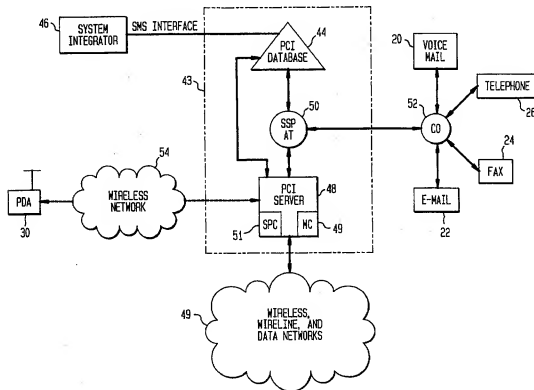
- [63] **Continuation of Ser. No. 309,336, Sep. 14, 1994.**
- [51] **Int. Cl.⁶** H04M 11/00; H04M 1/64
- [52] **U.S. Cl.** 379/58; 379/67; 379/57
- [58] **Field of Search** 379/58, 59, 60,
 379/57, 93, 67; 370/58.1, 110.1

References Cited**U.S. PATENT DOCUMENTS**

5,008,926 4/1991 Mishli 379/89

ABSTRACT

A personal communications internetwork provides a personal communications internetwork providing a network subscriber with the ability to remotely control the receipt and delivery of wireless and wireline electronic text messages. The network operates as an interface between wireless and wireline networks. The subscriber's message receipt and delivery options are maintained in a database which the subscriber may access by wireless or wireline communications to update the options programmed in the database.

3 Claims, 21 Drawing Sheets

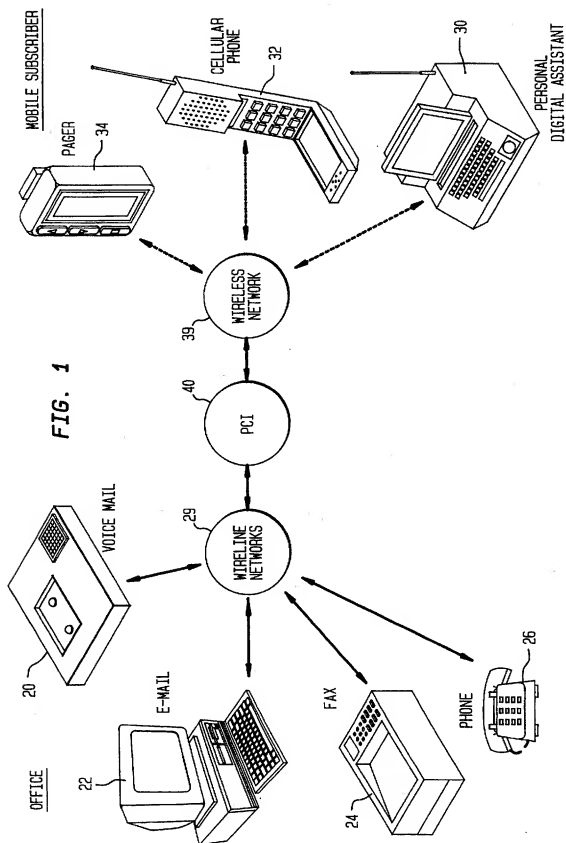


FIG. 2

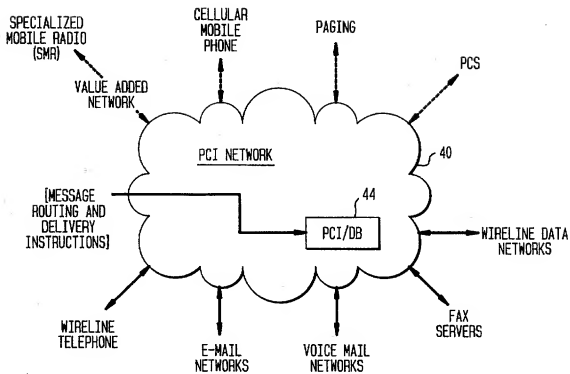
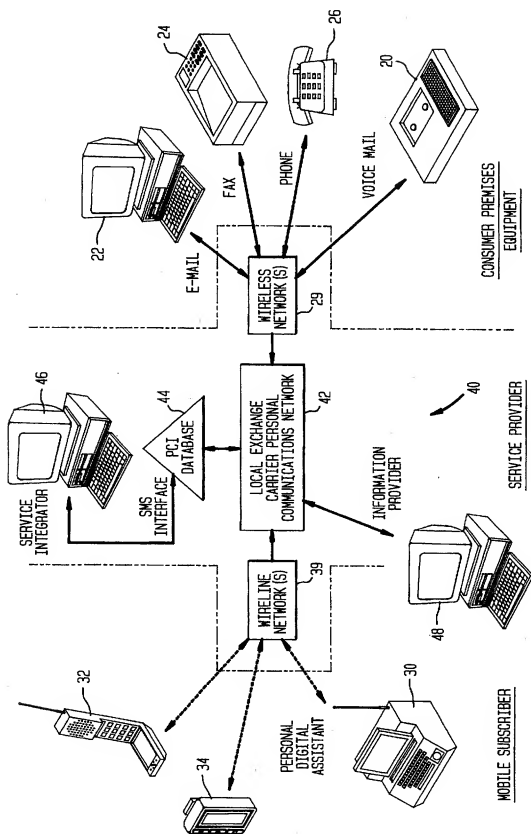
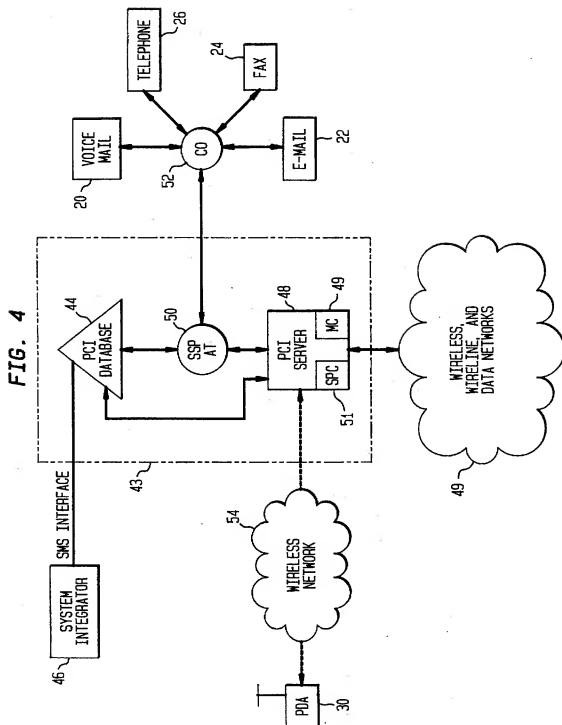
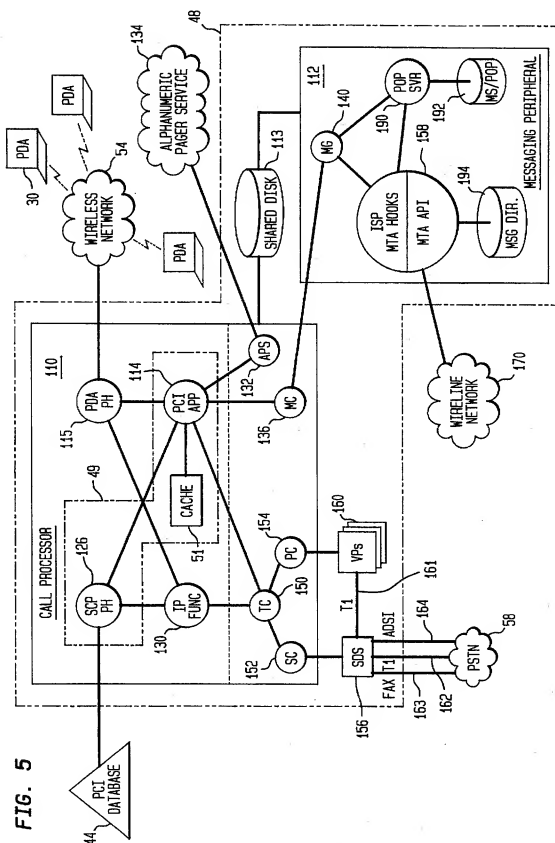


FIG. 3







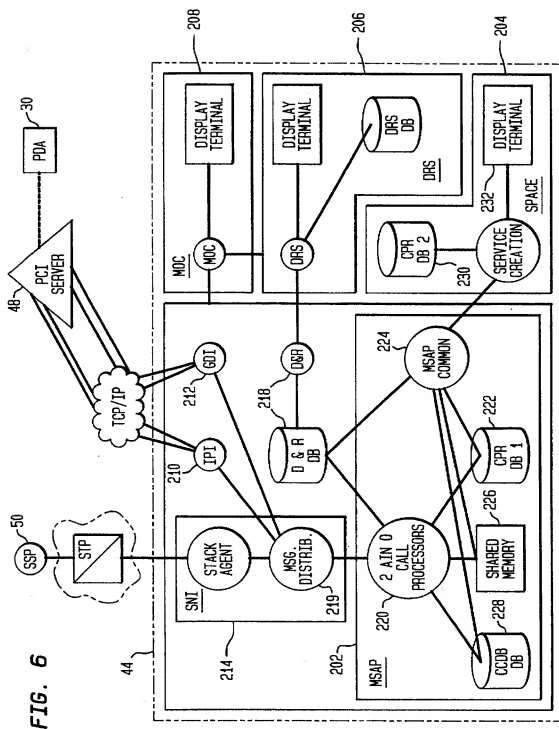


FIG. 7

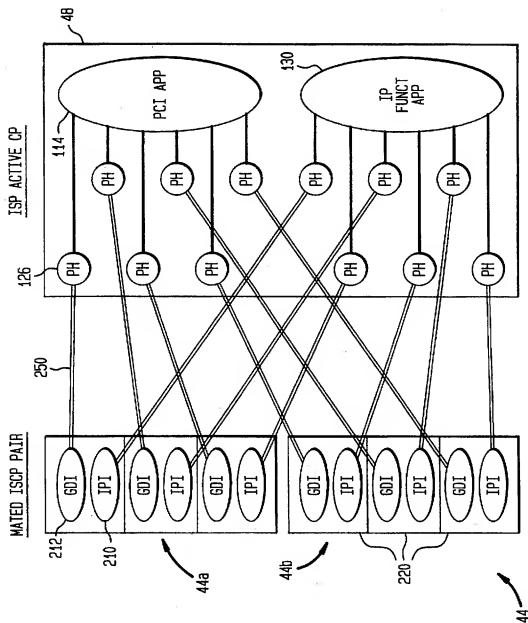


FIG. 8

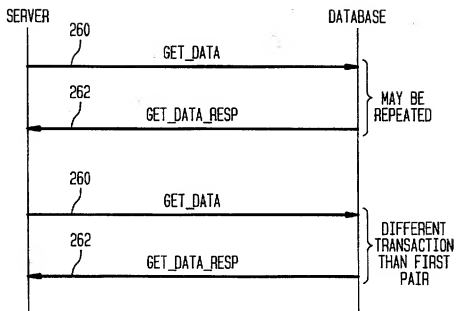


FIG. 9

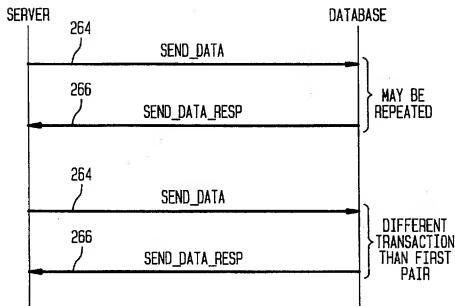


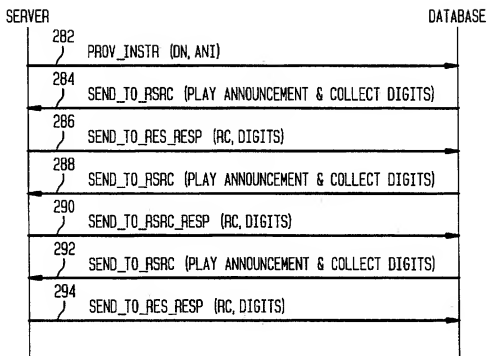
FIG. 10

FIG. 11

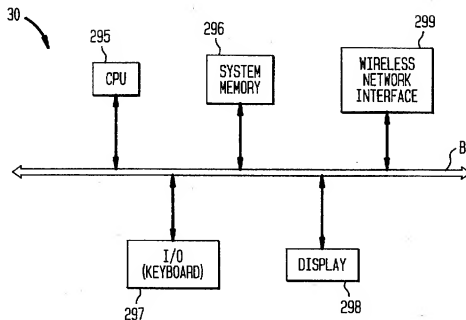


FIG. 12

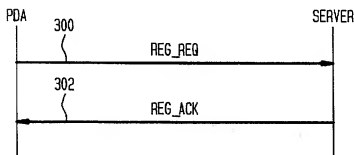


FIG. 13

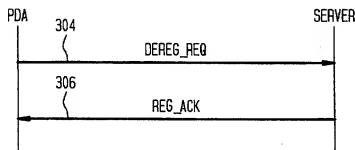


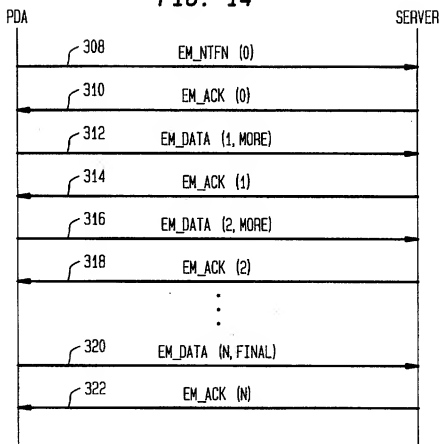
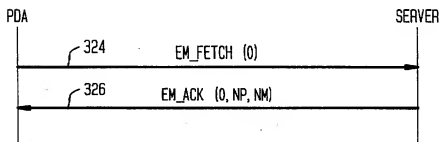
FIG. 14**FIG. 15A**

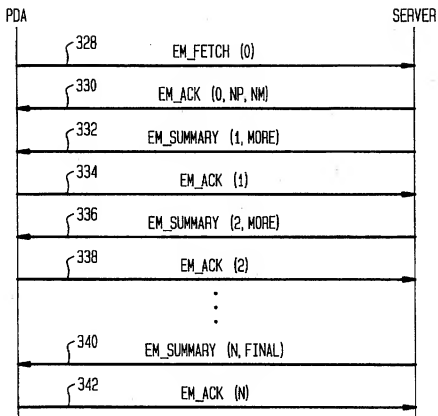
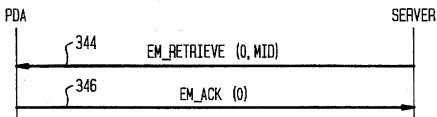
FIG. 15B**FIG. 16**

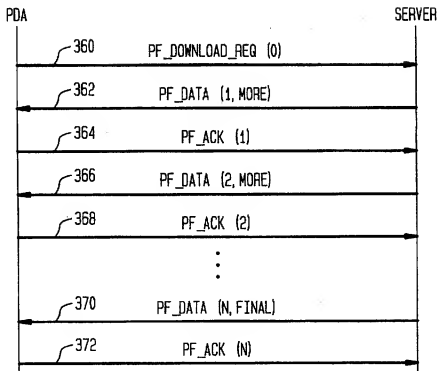
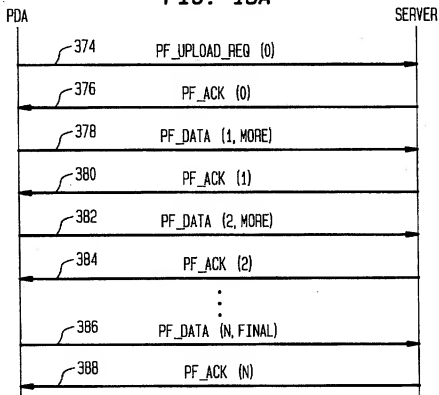
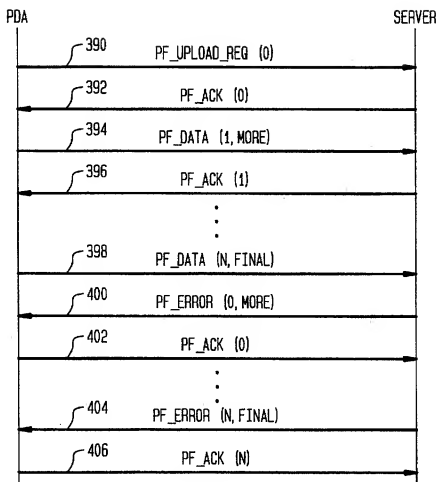
FIG. 17**FIG. 18A**

FIG. 18B

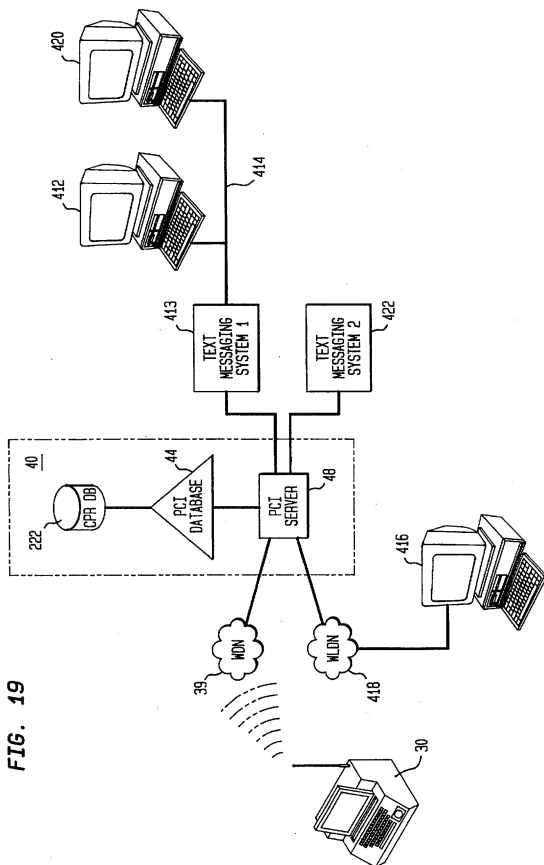


FIG. 20

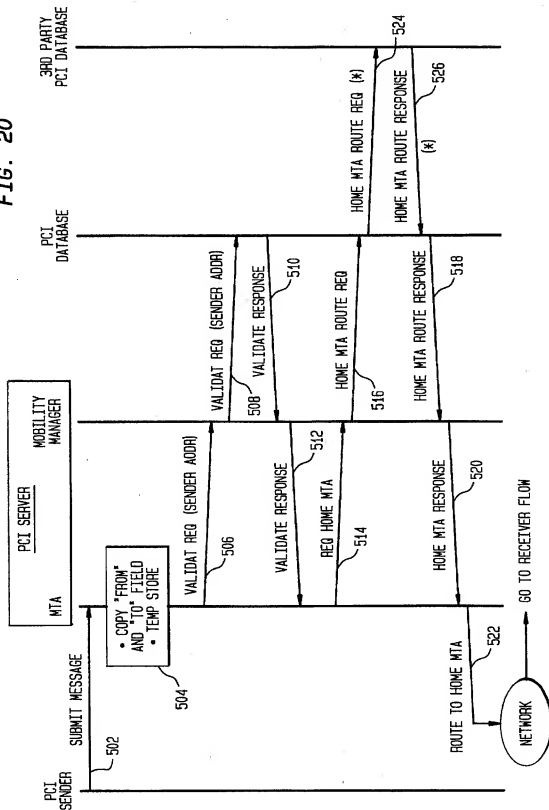


FIG. 21

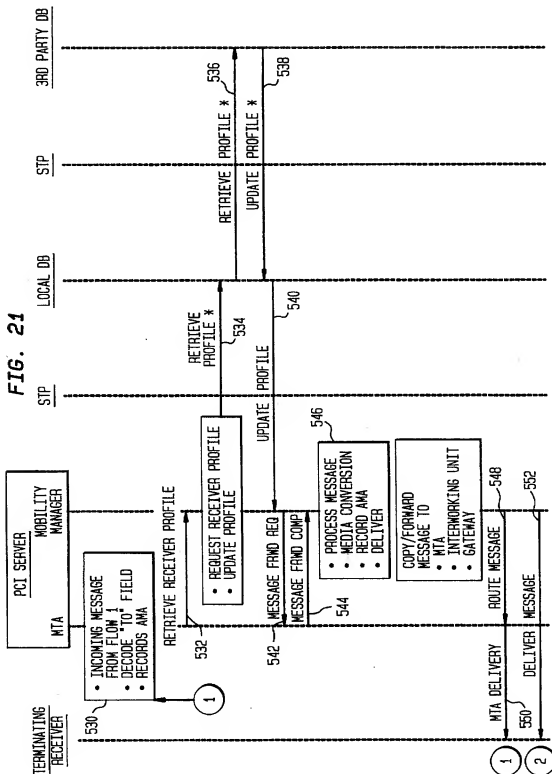


FIG. 22

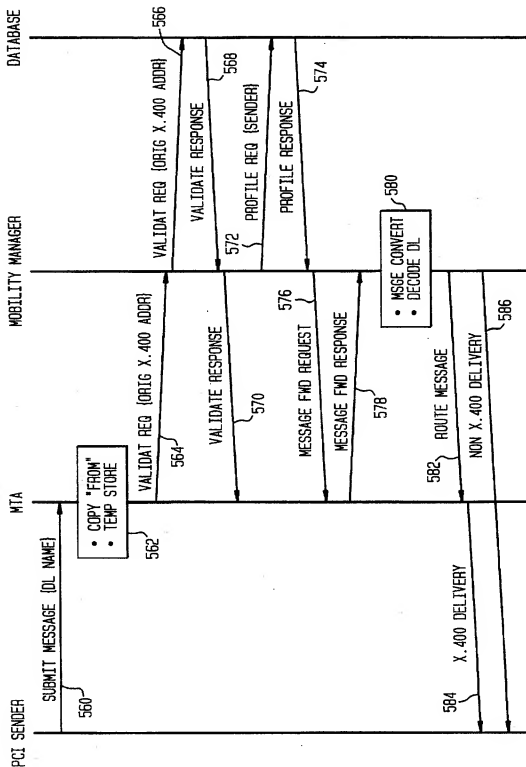


FIG. 23

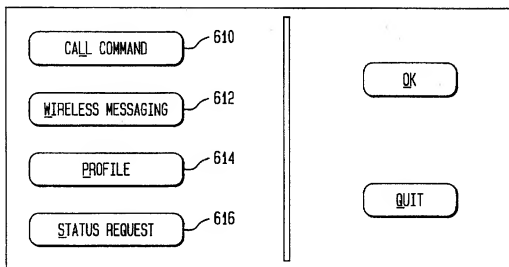


FIG. 24

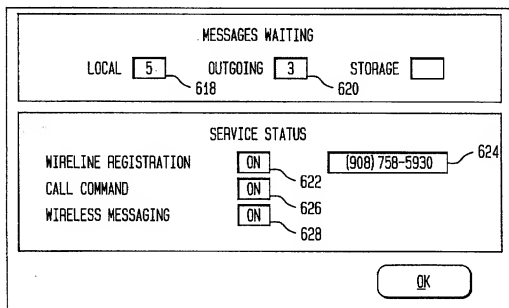


FIG. 25

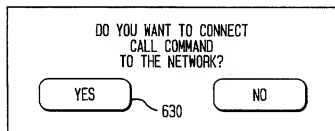


FIG. 26

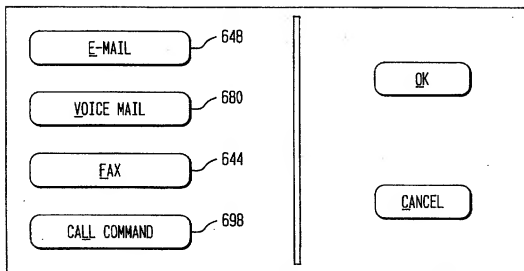


FIG. 27

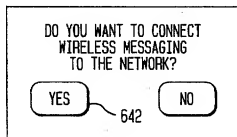


FIG. 28

DESTINATION SCREENING

MATCHED A MATCHED B NOT MATCHED

WIRELINE E-MAIL WIRELINE E-MAIL FAX

NOTIFICATION SCREENING

MATCHED NOT MATCHED OFF

FAX PAGER VOICE MAIL

SCREENING

ON OFF

SCREEN ON ADDRESSES

SCREEN ON SUBJECTS

OK

CANCEL

FIG. 29

INPUT E-MAIL ADDRESS:

cc! csm

cc! mpf1

cc! stantp

cc! kogut

cc! Pizzo

cc! rubin

cc! buckner

prefect! rnm1

OK

ADD TO LIST

DELETE

CANCEL

FIG. 30

SUBJECT 1

SUBJECT 2

SUBJECT 3

SUBJECT 4

SUBJECT 5

OK

CANCEL

ELECTRONIC MASSAGING NETWORK

RELATED CASE INFORMATION

This case is a continuation application of U.S. patent application Ser. No. 08/309,336 filed on Sep. 19, 1994. The contents of that application are incorporated herein by reference.

FIELD OF THE INVENTION

The present invention is directed to an internetwork for personal communications and, more particularly, to a network which provides a variety of electronic text delivery, receipt, and notification options.

BACKGROUND OF THE INVENTION

The use of messaging as a means of day-to-day communications continues to grow and evolve, particularly in a business context. Messaging includes electronic mail (e-mail), facsimile transmissions (fax), paging, voice mail, and telephone communications. The introduction of the cellular phone and other wireless communications facilitated the advent of the "mobile office". The mobile office allows an employee, for example, to work away from the office on a portable computer and be in constant touch with the office via a cellular phone.

The messaging options described above are available to businesses of all sizes, as well as individual users, from a variety of service providers. Many offices have some or all of the messaging options described above. The office may have certain messaging equipment (referred to as "consumer premises equipment" or "CPE") connected to one or more wireline networks. That is, the office may have telephones, fax servers, and voice mail systems connected to phone lines, and computers having modems for e-mail connected to packet networks which are connected via phone lines. The mobile employee may have certain wireless messaging equipment, such as a pager, a cellular telephone, or a personal digital assistant ("PDA"), which is typically a notebook computer connected to a wireless communication network.

One important goal of personal communication services is to allow users to communicate from anywhere to anywhere at any time. Such personal communication services generally involve multiple service providers including local and long distance telephone companies and cellular telephone companies. An example of a personal communication service is as follows:

A personal communication service provider (e.g., a cellular telephone company) enables traveling users to rent a wireless portable phone from a rental phone company (e.g., from an airline or car rental company). Using the rental phone, the user is provided with basic mobile phone service from the personal communication service provider. In addition, the user would like the following features:

- 1) The user wants calls directed to his/her office or home to be automatically forwarded to the rental portable phone, without informing anyone that he/she is traveling.
- 2) To avoid unimportant incoming calls (and corresponding incoming call charges), the user would like to restrict the number of people who can call the rented portable phone.
- 3) It is important to the user that the rental phone features be activated instantly, so that calls can be made immediately upon the user's arrival at the visiting location.

This kind of personal communication service involves a plurality of service providers. These providers are (a) the

local telephone company at the home location, (b) a long distance telephone company, (c) the local telephone company at the visiting location, and (d) the personal communication service provider (i.e., the cellular telephone company) at the visiting location. All of these are referred to herein as "service providers".

To enable this kind of personal communication service, involving multiple service providers, interoperability problems among the different service providers must be resolved. The interoperability problems can be divided into two categories: (a) location tracking and (b) service management.

The interoperability problem for location tracking has been addressed by adopting signaling protocols used by the mobile phone industry. Location tracking functions are implemented using two location registers. One of the registers, maintained by the local telephone company of the user's home location, is called the Home Location Register (HLR). The other register, maintained by the local telephone company of the visiting location, is called the Visiting Location Register (VLR). The HLR stores customer profile data and the location of the VLR of the user. The customer profile data contains important information such as the user's name, address, preferred long distance carrier, service features (e.g., call forwarding and call restriction), billing, and other administrative related information. When the user travels to a new visiting location, a new VLR is created in the new location. A part of the profile data stored in the HLR is transmitted and loaded into the VLR such that the service provider at the visiting location can implement service features for the visiting user. When the user travels to a new visiting location the location of the VLR stored in the HLR is changed to the new VLR location, and the VLR in the previously visited location is deleted. The process of creating a new VLR, loading profile data to the VLR, and updating the visiting location of a user in the HLR is called "automatic roamer registration".

The interoperability problem for service management is much more complex than that for location tracking. Service management refers to a collection of functions required to enable a personal communication service user to subscribe to, modify, and activate service features anywhere and at any time. Examples of service management functions include phone number administration, customer profile data management, service activation, and security administration. The phone number administration function is important for maintaining the uniqueness of phone numbers. The customer profile data management function provides customer profile databases and user interfaces for creating, modifying, or transferring such databases. The service activation function extracts part of the data specifying service features from the profile data and loads this data into physical communication systems that process calls. The service activation function also controls the activation and deactivation of the service features. The security administration function prevents or detects unauthorized uses of services and service management functions.

Service management functions of this type are needed to provide personal communication services involving multiple service providers. Such service management functions generally require interactions between application software and various databases owned and operated by the different service providers. Consider an application which enables a nomadic user to subscribe to a personal communication service from any service provider at any location. An example of such a service is call forwarding to a temporarily rented portable phone. The application may, for example,

need to perform the following database access operations at databases maintained by various different service providers: check credit databases owned by credit card companies or phone companies to determine whether the user is able to pay for the service; check the customer profile database in the user's HLR to determine whether the user is currently located in a place other than the visiting location currently stored in the HLR; check the credit and network databases of long distance phone companies specified by the user to determine whether the user can use a particular long distance carrier in the visiting location; load profile data into the VLR at the visiting location and update the HLR with the location of the VLR if necessary; and load the profile data to the call processing systems and activate the service.

The user may need to send or receive messages from any or all of the messaging options described above at a visiting location. That is, the user may want to receive or receive notification of e-mail, faxes, phone calls, or voice mail at a visiting location or to send e-mail or faxes from a wireless terminal. The need to integrate these various types of messaging options and to interconnect the many service providers has, until now, been largely unaddressed.

It is also desirable for the mobile employee to be able to limit the messages sent to the wireless messaging equipment, so that only urgent messages are received when away from the office and unwanted incoming calls are avoided. The mobile employee may also wish to route certain incoming wireless messages and phone calls to other destinations, such as an office fax machine or a colleague's telephone.

Therefore, it is an object of the present invention to provide a mobile service subscriber the ability to remotely control the addressability, routing, accessibility, and delivery of messaging options.

It is another object of the present invention to provide an internetwork which interconnects messaging services with both wireless and wireline networks.

It is yet a further object of the invention to provide a control over the messages routed to wireless messaging options.

SUMMARY OF THE INVENTION

These objects are obtained by a personal communications internetwork providing a network subscriber with the ability to remotely control the receipt and delivery of wireless and wireline electronic text ("e-mail") messages. The network operates as an interface between wireless and wireline networks. The subscriber's message receipt and delivery options are maintained in a database which the subscriber may access by wireless or wireline communications to update the options programmed in the database.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects and features of the invention will become apparent from the following drawings, wherein:

FIG. 1, 2, and 3 are overviews of the PCI networks;

FIG. 4 is an overview of one node of the PCI network according to the present invention;

FIG. 5 is a block diagram of an exemplary PCI server according to the present invention;

FIG. 6 is a block diagram of an exemplary embodiment of a PCI database according to the present invention;

FIG. 7 is a block diagram of the logical connections between the PCI server and PCI database according to the present invention;

FIGS. 8, 9, and 10 illustrate exemplary message flows between a server and a database according to the present invention;

FIG. 11 is a block diagram of a personal digital assistant according to the present invention;

FIGS. 12, 13, 14, 15a, 15b, 16, 17, 18a and 18b illustrate exemplary message flows between a PDA and PCI server;

FIG. 19 is a block diagram of a text messaging portion of a PCI network;

FIGS. 20, 21, and 22 illustrate exemplary message flows in the PCI network; and

FIGS. 23, 24, 25, 26, 27, 28, 29, and 30 illustrate exemplary screens displayed to a PCI subscriber using a wireless PDA.

DETAILED DESCRIPTIONS OF PREFERRED EMBODIMENTS

For clarity of presentation, the detailed description is set out in the following subsections:

I. PCI Overview

The overall network is illustrated in FIGS. 1-4. The network is an interface between a plurality of wireless and wireline networks, providing a subscriber with a variety of wireless and wireline message and voice delivery and receipt options.

II. The PCI Server

The PCI Server is illustrated in FIG. 5. The PCI server is a peripheral which performs messaging and call redirection functions and interfaces with the PCI database to update the subscriber profile.

III. The PCI Database

The PCI Database is illustrated in FIG. 6. The PCI database maintains the subscriber profile, controls CallCommand functions, and handles DTMF-based subscriber profile updates.

IV. The Server/Database Interface

The Server/Database interface is illustrated in FIGS. 7-10. The PCI server/PCI database interface provides for the transfer of information regarding the subscriber profile.

V. The PDA/PCI Interface

The PDA/PCI interface is illustrated in FIGS. 11-18. The PDA/PCI interface provides for the transfer of information between a remote wireless subscriber and the PCI.

VI. E-mail Messaging Services

E-mail messaging is the PCI in illustrated in FIG. 19. The PCI network provides the subscriber with a variety of e-mail delivery, receipt, and notification options, including screening and selective destination delivery of incoming e-mail.

VII. Message Flows

Certain message flows for wireless messaging in the PCI are illustrated in FIGS. 20-22. The three message flows illustrated are sending a message from one subscriber to another, receiving a message regardless of whether the subscriber is using a wireless or wireline terminal, and sending a message to a non-subscriber.

VIII. The PDA Application

The application residing in the PDA is described in FIGS. 23-30, which illustrate exemplary screens displayed to a PCI subscriber using a wireless PDA.

IX. Billing

Billing procedures for a PCI network use is briefly described.

X. Conclusion

A glossary of acronyms used in this specification is attached as Appendix A.

I. PCI Overview

FIG. 1 is a simplified overview of a personal communications internetworking ("PCI") according to the present invention. A consumer, an office for example, has various messaging equipment, such as a voice mail system 20, an e-mail terminal 22, fax machines 24, and telephones 26. These are all connected to wireline networks 29. For example, the e-mail terminal 22 may be connected to a data packet network, such as Internet, whose packets are carried over phone lines. The fax 24, phone 26, and voicemail system 20 may be connected to a Public Switched Telephone Network (PSTN), part of which belongs to a particular local phone service company, and part of which belongs to a particular long distance service provider.

A mobile communications subscriber (for example an employee who works at the office described above and travels frequently) has various portable messaging equipment, such as a PDA 30, a cellular phone 32, and a pager 34. These are connected to wireless networks 39. These wireless messaging options may be provided by different service providers. That is, the cellular phone may be connected to a wireless network of a cellular phone service provider, the pager may be connected to a different wireless network maintained by a pager service provider, and the PDA may be connected to a third wireless communications network maintained by yet another service provider.

A Personal Communications Interservice ("PCI") 40 according to the present invention is connected between the wireless 39 and wireline networks 29. The PCI 40 permits the mobile communications subscriber to send and receive messages between disparate networks and messaging systems and a variety of service providers. The mobile communications subscriber can receive e-mail, fax, pages, and voice messages under a single phone number while using either a wireless or wireline network. The subscriber may also select the media format and serving network used to receive messages. The subscriber may also select cross-media notification of incoming messages, (i.e., the subscriber may receive notification from a pager message that a voice mail message was received).

The subscriber selects the wireline or wireless network and media format to be used for delivering messages or notification of message receipt. The PCI 40 will perform a media conversion to allow, for instance, an e-mail message to be delivered to a fax server. The PCI 40 may also include accessibility controls which allow the user to screen messages by selected criteria such as media type (e.g., e-mail, fax, etc.), message length (e.g., voice mail messages less than three minutes), or sender (e.g., only messages from the office and a certain client are to be forwarded).

For example, the subscriber may have notification of a voice mail or fax message receipt directed to a wireless PDA in the form of e-mail messages. If the subscriber's wireless PDA is not turned on or otherwise not operating, the notification may be routed to an alternate wireless or wireline network. Notification to the subscriber that a voice mail message was received may be, for example, rerouted to the subscriber's pager, and notification that a fax has been received may be rerouted to the wireline e-mail.

FIG. 2 is a simplified version of the interconnections between various messaging systems and a PCI. As shown in FIG. 2, a subscriber provides the network with message routing and delivery instructions. These instructions are

received by a PCI database 44 and stored in a "subscriber profile" for that subscriber. This database controls the delivery of outgoing messages and the routing of incoming messages and message notification. (In FIG. 2, wireline communications are indicated with solid line connections and wireless communications are indicated with dashed line connections. The instructions to the PCI are shown with a solid line, but as will be explained in greater detail below, the instructions may be sent either by a wireline or wireless network.)

The PCI database 44 supports access to information authenticating the subscriber's identity and validating the types of services subscribed to, the subscriber's message delivery (incoming messages) options and origination (outgoing messages) options and voice (telephone call and voice mail) options. For origination, the subscriber may select message distribution lists with specific media delivery options. The database 44 also supports access to the portions of the subscriber profile that the subscriber may control.

The subscriber may use a personal telephone number to register at alternate wireline and wireless terminals while maintaining use of the message screening and delivery options selected and stored in a subscriber's profile. This is called "personal mobility". Information about the location of a wireless or wireline network location to which the subscriber's terminal is connected automatically registers and deregisters a subscriber's terminal. This is called "terminal mobility".

FIG. 3 shows the PCI 40. The CPE (voice mail 20, e-mail 22, fax 24, and phone 26) are connected to wireline networks 29. The mobile subscriber equipment (PDA 30, cellular phone 32, and pager 34) are connected to wireless networks 39. Both the wireline and wireless networks 29, 39 are connected to a PCI 40 at a service provider. The networks 29, 39 are connected to a local exchange carrier (LEC) 42 for the personal communications internetworking.

A PCI database 44 is a physical communication system which provides call processing functions for a collection of central office switches. The PCI database 44 includes the mobile subscriber's profile, including message sending, message receiving, and service control options. The PCI database 44 may be a service control point (SCP) or a network adjunct. The PCI database may be connected via a service management system (SMS) interface to a service integrator 46. The service integrator 46 allows the service provider to update subscriber data and create and modify subscriber profiles.

The PCI database 44 preferably stores and updates the subscriber profiles. The profiles contain service related information for mapping services to subscribers (e.g., screening, routing, terminal selection by subscriber selected parameters, custom calling features, and the like); subscriber authentication data (e.g., password and user ID.); user status (registered or not registered); generic service profile for non-call associated service, such as subscriber address or social security number; specific profile for a non-call service (based on subscriber selected parameters); wireless data providers identification (e.g., what cellular phone provider is used); and specific profile for call associated services (e.g., call forwarding), based on user selected parameters.

FIG. 4 is a more detailed depiction of the one node 43 of the PCI. The PCI has a plurality of nodes and is preferably built on the Advanced Intelligent Network (AIN) architecture. Other network architectures may be used, but for illustrative purposes, the description is directed to an AIN-based network.

API server 48 is a peripheral which performs messaging and call redirection functions and interfaces with the PCI

database 44 to update the subscriber profile. The PCI server may be an AIN Intelligent Peripheral, such as a Bellcore Intelligent Services Peripheral, or a network adjunct. The PCI server is connected to a switch 50. In the AIN architecture, this switch is a Service Switching Point Access Tandem (SSP AT), but may be any suitable switch, depending on the architecture. The SSP AT 50 connects wireline networks to the CPE. The SSP AT 50 also connects the PCI server 48 with a central office (CO) 52. The SSP AT 50 also connects to the SCP 44. The SCP 44 and the PCI server 48 are directly connected. The LEC of FIG. 3 is part of a large network and includes the PCI database 44, the PCI server 48, and the SSP AT 50. The PCI database may be connected to an SMS interface to a system integrator 46, as described above.

The PCI server 48 is also connected to various wireless and wireline networks 49 via signaling connections in these networks to transmit and receive information for all of the messaging options. Illustratively, the PCI server provides access to Public Packet Switched Networks (PPSN), Public Switched Telephone Network (PSTN), Integrated Signaling Digital Networks (ISDN), X.25 networks and TCP/IP networks and may include access to asynchronous transfer mode (ATM), Switched Multimegabit Digital Service (SMDS), and Frame Relay networks.

The mobile subscriber may access his or her subscriber profile to change message sending, message receiving, and service control options. These option changes are sent to the PCI database 44 to be stored in the subscriber profile FIG. 4 shows, for example, a PDA 30 connected to the PCI server 48 by a wireless network, but the subscriber may also use wireline e-mail, or wireless or wireline telephones (using DTMF signals) to access the subscriber profile. The messages from the PDA, for example, are sent by a wireless network 54 to the PCI server 48 using, for example, an X.25 transport.

Delivering PCI service to a subscriber who may be present on a number of different systems requires storage, movement and caching of the service profile associated with that subscriber. A mobility controller 49, located in the PCI server 48, is a controller and data store, which dynamically maintains service control information for a Message Transfer Agent (MTA), described below, in the PCI server 48, which connects the PCI server 48 to wireless data networks.

Data storage functions are handled by two tiered entities. The subscriber profile is preferably located in the PCI database 44 and is the top of the hierarchy where permanent records such as service profile, authentication and validation information, and the like of the subscriber or device are maintained and performing status and location management and mapping are performed. A service profile cache 51 is preferably located in the PCI server 48 and is a local cache entity which stores on a "needs basis" information such as service profiles and validation status and maintains a local repository for the service recipient. It also administers information necessary to serve the wireless data network entity, as well as sending updates to the permanent storage entity PCI database. The service profile cache 51 maintains the personal data associated with the processing of the mobility controller 49. The mobility controller 49 interacts with the PCI database-based subscriber profile (or third party database) on behalf of the cache to obtain service profiles and location information related to wireless terminals.

PCI may also provide directory services as a value-added component. The X.400 MTA can query a local directory serving agent in the PCI server 48 for addressing and routing

information. If the information is not local, the PCI server 48 will need to get the addressing information from another PCI server 48 at another PCI node or an interconnected private directory serving agent which maintains a separate information base. By using the existing standard, the PCI network and mail PCI servers message handling can independently manage the networks without interfering with the PCI service.

II. The PCI Server

The PCI server is a peripheral which performs messaging and call redirection functions and interfaces with the service Control Point to update the subscriber profile. The PCI server performs a variety of functions. For example, an illustrative PCI server:

- is an X.400 Gateway;
- routes messages using the X.400 messaging protocol;
- connects proprietary messaging protocols into X.400 protocol;
- interfaces with wireless data networks;
- interfaces with messaging systems;
- interfaces with the PCI database to access subscriber profiles information;
- processes messages as specified by the user in the service profile;
- provides media conversion such as text to fax or fax to text;
- provides access to an X.500 directory to determine addressing schemes for packet data;
- supports signaling between wireless data networks for management functions such as registration; and
- maintains a service profile cache.

FIG. 5 is a detailed illustration of a preferred embodiment of a PCI server 48 according to the present invention. The PCI server 48 includes three main elements: a call processor 110, a data messaging peripheral 112, and a shared disk memory 113.

The call processor 110 comprises a plurality of interconnected computers. The messaging peripheral 112 may be implemented by a computer such as a DEC XAP system.

The call processor 110 includes a PCI applications server 114. The application server 114 is the central decision making point of the wireless messaging service described below in Section VI. Thus, the server 114 controls message routing, screening, and notification for the wireless messaging service.

The application server 114 is connected to a PDA protocol handler 115. The protocol handler is the interface to the wireless network 54, for example the RAM wireless network. This handles messages to be sent to and from the subscribers PDA 30. A plurality of personal digital assistants (PDA) 30 are connected to the wireless network 54.

The application server 114 also manages a PCI database protocol handler 126. The protocol handler 126 is the interface between the call processor 110 and the PCI database 44. The application server 114 also manages a Service Profile Cache (SPC) 51. The SPC 51 is maintained in the memory of the application server 114. The SPC 51 stores a subset of the data in the subscriber profile stored in the PCI database 44. This subset is subscriber profile information which currently needs to be accessed frequently by the PCI server 48.

The SPC 51 stores and accesses data related to access systems such as wireless data providers and messaging services, and subscriber location. The SPC 51 may store and update data related to the subscriber location such as routing

address for subscribers specific wireless terminals; store and updates services related data for a particular terminal type (such as uni- or bi-direction); maintain a list of the subscribers wireless data provides and message services; track the subscribers terminal status (registered or not registered); provide a generic service profile for non-call messaging service; and provide a specific profile for a non-call associated service based on subscriber selected parameters.

The application server 114 also manages the registration status of each application on each PDA 30 and controls customer profile information via each PDA 30.

The call processor 110 also includes an IP Functions Server 130. The IP Function Server 130 manages CallCommand applications. This server is also connected to the PCI database protocol handler 126 for communication with the PCI database 44 and the PDA protocol handler 115 for communication with the wireless network 54. The PCI database protocol handler 126 handles both interfaces between the PCI database and the PCI server, as described below.

Thus, the two main application servers in the call processor 110 are the IP Function server 130 for CallCommand applications and the PCI applications server 114 for wireless messaging services.

The call processor 110 also includes a plurality of communication interfaces. The protocol handlers 115 and 126 have already been discussed. The alphanumeric paging server (APS) 132 gives the call processor 110 the ability to provide alphanumeric paging services. The APS 132 includes one or more modems to communicate with terminal equipment of a network 134 maintained by a paging service provider. The APS communicates with the paging service provider using, for example, the TAP protocol (Telocator Alphanumeric Protocol).

The call processor 110 also includes a plurality of control processes which control peripheral equipment external to the call processor 110. These controllers are as follows:

A message controller 112 controls the data messaging peripheral 136 and controls the sending of messages between the call processor 110 and the data peripheral 112.

The mobility controller 49 comprises the PCI database protocol handler 126, the IP function server 130, the service profile cache 51, and the PCI application server 114. The mobility manager provides control logic for user authentication, service request validation, location management, user access to service profile, access registration, and communication management such as routing to user-specified destinations. The mobility controller 49 contains the service logic and handles service related processing for personal data and service access such as service feature analysis; access system mapping relationship information; identity management; subscriber validation and authentication; billing information based on the subscriber; wireless data specific routing information for message delivery and subscriber paging; subscriber service validation; and subscriber review and modification of the subscriber's profile.

A transaction controller 150 controls a switch controller 152 and a voice peripheral controller 154. The switch controller 152 controls the digital switch 156 which connects to the public switched telephone network 58. The voice peripheral controller 154 controls the voice peripherals 160, which are for example text-to-speech converters.

The switch 156 and the voice peripheral 160 are also connected by a T1 line 161. The digital switch 156 is connected to the public switched telephone network by a plurality of transmission media such as T1 lines 162, fax lines 163, and ADSI lines 164.

The data messaging peripheral 112, which is optional, is now discussed in greater detail. The data messaging peripheral is the gateway to the wireline electronic mail network, which network is designated 170. The data messaging peripheral has a message transfer agent 158 for transferring messages between the call processor 110 and the data networks 170, 54 either directly or through the PDA protocol handler 115. The messaging peripheral 112 also includes a POP (post office protocol) server 190 and associated memory 192 for providing a message storing capability. The message directory 194 is used for storing a subset of service profile cache 51 relating to the routing of e-mail messages.

The messaging peripheral 112 includes the message gateway 140. The message gateway 140 has the following capabilities:

- 1) Notifying the PCI application server 114 in the call processor that e-mail has arrived from the wireline e-mail network 170 for a subscriber.
- 2) Accept a request from the PCI application server 114 to send an e-mail message to a wireline address.
- 3) Accept a request from the application server 114 to provide all unread messages stored in the server 190 which would have been sent to a primary destination if the subscriber had been registered.
- 4) Accept a request from the application processor 114 to rewrite to the message store server 190 or back to the sender.

Using the call processor 110 and its associated peripherals, a wide variety of services may be performed. These have been discussed above briefly. However, to understand how the call processor 110 operates to provide these services, an exemplary description is provided.

For example, when a wireline e-mail message arrives at the PCI server's Data Messaging Peripheral 112, the messaging gateway 140 and messaging controller 136 send notification to the PCI application server 114 of the e-mail arrival. The PCI application server 114 will query the profile cache 51, or if necessary, the PCI database 44. Driven by data in the subscriber's profile, the PCI application server 114 executes service logic to determine where to forward the e-mail (i.e., forward to PDA 30 or to POP server 190 depending on screening outcome), and what media, if any, to use to send notification of the e-mail arrival.

If a subscriber wishes to update the subscriber profile by DTMF, the procedure is as follows. A call arrives at the PCI server 48. The switch controller 152 and transaction controller 150 forward the call to the IP functions server 130 based on the dialed number. The IP functions server 130 sends a provide_instructions 1129+message to the PCI database 44 to determine how to handle the call. The PCI database 44 sends a request to play an announcement and collect digits ("please enter PIN", collect PIN). The IP functions server 130 returns the result of this request to the PCI database 44. Again the PCI database 44 sends a request to the IP functions server 130 to play an announcement and collect digits ("voice menu", menu selection). The IP functions server 130 returns the result of this request to the PCI database 44.

This process repeats as users are guided through menus and change profile elements. The PCI database 44 interprets the collected DTMF tones and updates the subscriber's profile accordingly.

When a PDA 30 sends an e-mail message addressed to a wireline address the procedure is as follows. The PDA 30 sends a UDP send_mail message to the PCI application server 114. The PCI application server 114 detects the message is not destined for another PCI subscriber and

forwards the request to the messaging controller 136, which forwards it to the messaging gateway 140 which is in the Data and Messaging Peripheral 112. The messaging gateway 140 interfaces with the MTA 158 to send the e-mail to the wireline network 170 using, for example, the Simple Messaging Transfer Protocol (SMTP).

The PCI server 48 may be based, for example, on either an X.400 MTA or an SMTP router and can convert between both protocols. The PCI server 48 may receive text messages from a variety of different text messaging systems such as Internet mail, third party messaging systems, or proprietary messaging systems. In the example where PCI routes messages using an X.400 MTA, these messages must be converted to conform with X.400 protocol before they can be routed. Thus, an exemplary messaging gateway is an X.400 gateway, which can be designed and built by a person of ordinary skill in the art.

III. The PCI Database

A PCI Database 44 maintains the subscriber profile, controls the CallCommand functions, and handles DTMF-based subscriber profile updates.

The PCI database architecture shown in FIG. 6 comprises several application and support components. The application components include Multiple Services Application Platform (MSAP) 202; Service Provisioning and Creation Environment (SPACE) 204; and Data and Report Subsystem (DRS) 206.

The service components include the Maintenance and Operation Console (MOC) 208; the Intelligence Peripheral Interface (IPI) 210; the Generic Data Interface (GDI) 212; the Service Network Interface (SNI) 214; and the Data and Report Database (D&R) 218.

The service network interface (SNI) 214 provides a communication interface to external systems such as switch 50 and PCI server 48. These interfaces include the IPI 210 and GDI 212 which connect the PCI database to the PCI server via the TCP/IP network 213. The GDI 212 is used for uploading and downloading a subscriber profile to the PCI server 48. The IPI 210 is used for transmitting DTMF commands from a user via the PCI server 48. For redundancy, each intelligent peripheral interface (IPI) and generic data interface (GDI) processor preferably requires two logical connections to the PCI server.

The Multiple Services Application Platform (MSAP) 202 includes a call processor 220, a first call process request (CPR) database 222, an MSAP common 224, a shared memory 226, and a call contact database (CCDB) 228. The call processor 220 receives messages from and sends messages to a message distributor 219 in the SNI 214. The message distributor determines whether the message received from the call processor 220 is to be sent to the IPI 210 or the GDI 212. The call processor receives messages from the message distributor and sends them to the first CPR database, the CCDB 228, and/or the shared memory 226. The first CPR database 222 stores the subscriber profiles. The MSAP 224 connects the first CPR database 222 with the second CPR 230, which resides in SPACE 204. MSAP common 224 updates one of the CPR databases 222, 230 when changes have been made to the other CPR database. The CCDB 228 is a temporary, dynamic storage for storing subscriber profiles, and related data during profile update procedures. The shared memory 226 allows different processors to use the same data.

SPACE 204 is a service provider-operated module through which new PCI database applications are created and new subscriber profiles are initiated. SPACE 206 includes the second CPR database 230 which contains the

identical information as the first CPR database 222 in MSAP 202. When a new subscriber profile is to be created, a service provider uses a display terminal 232 in SPACE to provision a new service profile including certain subscriber information. The subscriber profile is activated through MSAP when the user initially registers. Service provider changes made to the second CPR database 230 are transmitted to the first CPR database 222 in MSAP via the MSAP common 224. Changes made to the second CPR database 230 by a service provider are not transmitted to the service profile cache 51 in the PCI server 48 until a later time. That is, the PCI database 44 does not send data to the PCI server 48 unless requested by the server 48. The server profile cache 51 will be updated with this new information the next time the PCI server 48 requests a profile download, for instance when the subscriber next registers. SPACE 204 provides a function parallel to the Service Management System described above.

The Data and Report Subsystem (DRS) 206 collects data about the PCI database 44 usage which may be helpful to the service provider. For example, errors made by the subscriber when updating the user profile are noted. The types of alterations made, times such alterations are made, and the like are also stored for future use by the service provider.

MOC 110 is a network maintenance support system which monitors the status of the network and checks for system failures and the like.

When a subscriber wishes to update the subscriber profile using a PDA 30, the procedure is as follows. The PDA 40 communicates with the PCI server 48. The PCI server 48 sends a GetData message having a "Service Key", which is a preferably a ten digit PCI subscriber number (e.g., a telephone number), to the PCI database 44 over the GDI 212. The GDI 212 translates the GetData message into a format understandable by the PCI database 44. The message is sent through the message distributor 219 and call processor 220 to the first CPR database 222 where the subscriber profile resides. The Service Key is used to obtain the correct subscriber profile and the profile is sent through the call processor 220 to the message distributor 219. The message distributor determines that this message is to be sent to the PCI server 48 via the GDI 212. (The reason for this is discussed below.) The GDI 212 translates the data into a format suitable for the TCP/IP network and is transmitted to the PCI server 48. The requested changes are performed in the PCI server 48 and the updated profile is sent back to the PCI database 44 through the TCP/IP network, the GDI 212, message distributor 219, call processor 220 and to the first CPR database 222. The call processor 220 also sends a message through the GDI 212 to the PCI server 48 which will be sent a wireless transmission to the PDA 30 acknowledging the subscriber profile update. The changes are also sent to the MSAP common 222 where they are sent to the second CPR database 230 in SPACE 204.

During this process, information may be temporarily stored in the Call Contact Database (CCDB) 228. The CCDB database 228 provides temporary storage for subscriber profile updates that are suspended because they are waiting for action by a subscriber or waiting for data from an external system, such as the PCI server 48. During the time intervals between action by the user or delays in receiving data from an external system, the call processor 220 stores the information in the CCDB database 228 and processes other calls.

When a subscriber desires to update his or her subscriber profile using a touch tone phone, the procedure is as follows. The subscriber calls, for example, a service number pro-

13

vided by the service provider. The call is routed to the PCI server 48. The PCI server 48 sends a message to the PCI database 44 via the IPI 210 that the DTMF commands are present. The message is sent through the message distributor 219 to the call processor 220. The appropriate subscriber profile is retrieved from the first CPR database 222 in the MSAP 202.

The call processor 220 instructs the PCI server 48 to play a voice announcement instructing the caller to enter the subscriber ID and password, by pressing the appropriate digits on the touch-tone phone. The information is entered by the caller, and the PCI database 44 validates this information. If the validation determines that the caller is an authorized subscriber, the PCI database 44 instructs the PCI server 48 to ask the subscriber to select which subscriber profile information is to be modified. Currently, only two fields are modifiable using DTMF messaging: changing a wireline registration or recording a personalized greeting. The subscriber selects either registering at a wireline phone or recording a personalized greeting. If wireline registration is selected, the PCI database 44 instructs the PCI server 48 to prompt a ten digit telephone number to which all incoming calls will be routed. If the subscriber selects to record a personalized greeting, the PCI database 44 instructs the PCI server 48 to prompt the subscriber for a new greeting.

If invalid information is entered at any time, the PCI server 48 plays an error message to the subscriber and the subscriber retries the modification. If the retry fails, the call is terminated. Otherwise, the subscriber's profile is updated according to the modification, data synchronizing the messages are sent to the PCI server 48 and the call processor 220 instructs the PCI server 48 to inform the subscriber that the PCI service profile was updated.

The call processor 220 also sends a message through the message distributor 219 to the GDI 212 and to the PCI server 48 which updates the service profile cache 51 in the PCI server 48. The changes stored back in the first CPR database 220 are sent to the MSAP common 224 where they are sent to the second CPR database 230. Note that DTMF function signals, which use the 1129+ protocol, are routed through the IPI 210 and the subscriber profile data, which uses the GDI protocol, are routed through the GDI 212.

IV. The PCI Server/Database Interface

The interface between the PCI server 48 and the PCI database 44 is based on two protocols. The first protocol is 1129+. This protocol will be used to support the PCI CallCommand feature and for subscriber initiated profile manipulation using DTMF. The second protocol is Generic Data Interface (GDI). The GDI is used for subscriber profile management, specifically downloading a subscriber profile from the PCI database 44 to the PCI server 48 and for applying updates to the profile stored in the PCI database 44.

FIG. 7 shows the logical links from the PCI database 44 to the PCI server 48. The PCI database 44 consists of a mated pair of PCI databases 44a, 44b, each containing three call processors 220 which each share the load. The links 250 are TCP/IP links between Intelligent Peripheral Interface (IPI) 210 and the Generic Data Interface (GDI) 212 processors on the PCI database 44 to the PCI server call processor. Two logical connections are made from each IPI 210 and GDI 212 processors to the PCI server for redundancy. Thus, a full SCP configuration supporting PCI would preferably require 24 logical links, as shown in FIG. 7. The PCI database initiates the opening of the logical links.

In this illustrative embodiment, the CallCommand feature employs the 1129+ protocol. For the wireless messaging feature, PCI uses the GDI protocol. The GDI tag IDs

14

assigned for the PCI subscriber profile elements are provided in Appendix B.

Appendix B also shows the PCI profile data, including the profile elements, their data types, maximum lengths, and GDI tag IDs. An * indicates elements which were shortened to 32 bytes because of GDI byte limitations. The description of the types and lengths of these elements is as follows:

dN BCD-encoded digits. The number N represents the maximum number of BCD digits, not octets.

cN Up to N ASCII characters.

bN Binary integer N bytes in length, in network byte order (highest order bit transmitted first).

Because the portion of the PCI subscriber profile downloaded to the PCI server is large (preferably approximately 1,000 bytes), and a maximum Transaction Capable Application Program (TCAP) message size is 256 bytes, the profile must be managed in segments. The service profile is divided into six segments as shown in Table 1. Each segment is assigned a unique numeric identifier.

PCI Profile Segment	Segment ID (decimal)
Personal data	1
CC service profile	2
e-mail routing	3
e-mail subject screening	4
e-mail from screening	5
Voice mail profile	6

Certain data in a subscriber profile provides a subscriber's preferred media for messages delivery and notification. The encoding for these types are given in Table 2.

Media Type	Code
Alphanumeric Pager	A
e-mail message store	S
Fax	F
PDA	P
Voice mail	V
Wireline e-mail	E
Null	Z

For example, if the subscriber prefers to receive e-mail which passes screening via the PDA 30, then the "primary destination one" profile element will contain a "P".

FIG. 8 illustrates a message flow for profile retrieval using the GDI protocol. A subscriber attempts to register with the PCI server either explicitly or implicitly (registration is discussed in detail below). The PCI server 48 send a GDI GetData query to the PCI database 44 over one of the GDI links (line 260). The PCI server 48 may send one GetData data query for each PCI profile segment. Each query will be processed by the PCI database 44 as an independent transaction with a unique TCAP transaction ID. Each GetData query sent by the PCI server 48 will include a "Service Key" parameter which is a ten-digit PCI subscriber number (e.g., a telephone number). This key should be used by the PCI database 44 to identify the subscriber. In each GetData is a list of tag IDs listed in the profile elements to be retrieved. The PCI database 44 responds to the GetData data query with a GetData response (line 262). The response contains a return code and data for each element requested in the GetData data query.

FIG. 9 provides a message flow between the PCI server 48 and the PCI database 44 for a profile update originating from a wireless PDA 30. This wireless profile update uses the GDI protocol. A subscriber performs a profile manipulation

activity, and the PDA 30 sends a profile data message to the PCI server 48. The PCI server 48 sends a GDI SendData query to the PCI database 44 over one of the GDI links (line 264). The PCI server 48 may send one SendData query for each PCI profile segment for which a profile element was updated. Each query will be processed by the PCI database 44 as an independent transaction with a unique TCAP transaction ID.

Each Send Data query sent by the PCI server 48 will include a "Service Key" parameter which is the ten digit PCI subscriber number. This key should be used by the PCI database 44 to identify the subscriber. Each SendData query contains a list of tag IDs provided in Appendix B and data for the profile elements to be updated. Not all tags in this segment may be included in the Send Data query; only those profile elements which are actually updated by the subscriber will be sent. The PCI database 44 should not update data for which no tag was included in the SendData query.

The PCI database 44 responds to the SendData query with a Send Data response (line 266). The response contains a return code for each element requested in the SendData query.

FIG. 10 is an illustrative example of one possible message flow between the PCI server 48 and the PCI database 44 for a DTMF profile manipulation message. The DTMF profile manipulator uses the 1129+protocol through the IPI 210. The exact call flow for DTMF profile manipulation depends upon the implementation of service logic by the service designer, and upon options selected by the PCI subscriber.

As shown in this illustrative example, when a call arrives at the PCI server, the PCI server sends an 1129+provide... instructions query to the PCI database (line 282). The called number contains a dialed number (i.e., the service number for a DTMF update), while the ANI (automatic number identification) field contains the ANI, if any. The PCI DTMF profile manipulations Call Process Request (CPR) is triggered by the dialed service number. The CPR 222 instructs the PCI server to play announcements and collect digits, guiding the subscriber through voice menus and prompts (lines 284, 288). The PCI server responds to each request with digits collected (lines 286, 290, 294). The CPR updates subscriber's profile with data collected via DTMF.

V. PDA/PCI Interface

Communication between the PDA and PCI use, for example, an X.25 transport using the UDP IP protocol. A brief discussion of the PDA structure is provided. The PDA 30 is preferably a notebook or palm top computer having a wireless network interface. The PDA may be, for example a Hewlett Packard Omnibook 300 notebook computer running a PCI application. FIG. 11 illustrates an exemplary PDA. The PDA 30 has a central processing unit 295 connected to a bus B. The central processing unit ("CPU") 295 performs most of the computing and logic functions of the PDA 30. A memory 296 is connected to the bus B, which stores information to be provided to the CPU 295 or otherwise used by the PDA 30. An input/output device 297, such as a keyboard, is also connected to the bus B which allows a user to input data for storage in memory 296 or for use by CPU 295. A display 298 is connected to the bus B. The PDA 30 also has a wireless communication interface 299 for communication with a wireless communication network.

The PDA/PCI interface involves six types of message flow. These messages are: (1) registration/deregistration; (2) wireless messaging; (3) retrieving e-mail; (4) cross-media notification; (5) CallCommand; and (6) profile management.

There are two types of registration and deregistration: explicit and implicit. Explicit registration occurs when a PCI

subscriber starts the PCI application software on the PDA 30 (this is called start-up registration) or when the subscriber clicks a status check button or one of the service registration request buttons on the PDA 30 either for the CallCommand or wireless messaging service. Once successfully registered, if the subscriber's profile is not already present in the service profile cache 51 maintained by the PCI server 48, the PCI server 48 will request a download of the subscriber's profile from the PCI database 44 to the service profile cache 51. The PCI server 48 sets the subscriber's registration status in the cache 51 to match those requested by the subscriber for the wireless messaging service for the call command service.

FIG. 12 illustrates one example of the message flow between the PDA 30 and PCI server 48 during explicit registration. This flow is also used by a subscriber to check registration of CallCommand or wireless messaging services. A subscriber starts the PCI application software on the PDA or clicks the service status check, CallCommand registration, or wireless messaging registration buttons on the PDA. The PDA sends a registration request to the PCI server 48 with the subscriber's validation information (subscriber ID and password) (line 300). The PDA 30 also starts a timer during which the PDA 30 will wait for a response from the PCI server 48. The PCI server 48 server receives the registration request and checks if the subscriber is provisioned and if the subscriber ID and password are correct. The PCI server then sends a registration acknowledgement (line 302). If the subscriber is not provisioned, no service profile exists and the acknowledgement includes an "unrecognized subscriber" response. If the subscriber ID and password are invalid, the acknowledgement includes an "incorrect password/PIN" response. Otherwise, the PCI server acknowledgement includes a "success" response. If the PDA 30 does not receive an acknowledgement from the PCI server within a predetermined time, it aborts the registration attempt and tells the subscriber to try again later.

Implicit registration automatically registers a subscriber for the wireless messaging service when the subscriber is currently not registered and wishes to send or fetch e-mail from or to a PDA 30. Implicit registration is done as follows. The PCI server receives a fetch or send request from a subscriber who is not registered for the wireless messaging service. The PCI server 48 retrieves a copy of the subscriber's service profile from the PCI database 44, if necessary, and validates the subscriber's ID and password. The PCI server 48 validates the profile contents to make sure that subscriber may use the wireless messaging service. If wireless messaging is permitted, the PCI server 48 processes the request. Otherwise, it sends an acknowledgement indicating the reason why the subscriber is not permitted to use the wireless messaging service. The message flow is the same as illustrated in FIG. 12.

Once the subscriber is registered for either the CallCommand service or the wireless messaging service, the subscriber remains registered until the subscriber explicitly deregisters by either quitting the application or clicking the deregistration button on the PDA 30. The subscriber can also be implicitly deregistered for the wireless messaging service by the PCI server 48 provided the PCI did not detect any wireless messaging activities to or from that subscriber for a given duration of time. Although the subscriber is deregistered, the subscriber's service profile will remain in the service profile cache 51. The profile remains in the cache as long as the PCI server has some activity for the subscriber, such as incoming e-mail messages within a predetermined time, such as four hours.

No PDA-to-PCI server messages may be sent by the subscriber to implicitly register for CallCommand. thus, a

subscriber should not be implicitly deregistered from this service. Implicit registration and deregistration occurs only for the wireless messaging service, and not for CallCommand. A subscriber remains registered for CallCommand as long as he or she is running the CallCommand software application on the PDA.

Explicit deregistration occurs when a subscriber quits the PCI application software on the PDA (this is called exit deregistration) or when the subscriber clicks one of the service deregistration request buttons on the PDA for the CallCommand or wireless messaging services. FIG. 13 is an illustrative embodiment of a message flow between the PDA 30 and PCI server 48 for explicit deregistration. A subscriber quits the PCI application software on the PDA or clicks a deregistration button on the PDA. The PDA 30 sends a deregistration request to the PCI server 48 with the subscriber's validation information (the subscriber ID and password) (line 304). The PDA 30 also starts a timer during which the PDA will wait for a response from the PCI server 48. The PCI server 48 sends an acknowledgement (line 306). The PCI server 48 receives the deregistration request and checks if the subscriber ID and password are correct. If the subscriber ID and password are not correct, the acknowledgement includes an "incorrect password/PIN" response. Otherwise, the acknowledgement includes a "success" response. If the PDA 30 does not receive an acknowledgement from the PCI server 48 after a predetermined time, the PDA 30 assumes that it is out of radio coverage and informs the subscriber to retry later.

Implicit deregistration occurs when the PCI does not detect any wireless messaging activity from or to the subscriber for a given duration of time, for example four hours. The PCI will also try to implicitly deregister a subscriber from the wireless messaging service in the middle of the night in the event that the subscriber inadvertently left the PDA 30 turned on. The PCI server 48 keeps a time-stamp of the most recent wireless messaging activity for each registered subscriber in the subscriber's service profile maintained in the service profile cache 51. Whenever the PCI server 48 detects any wireless messaging activities to or from a particular subscriber, the time-stamp is updated to the current time. The stored time-stamp of a registered subscriber is periodically compared to the current time. When a predetermined time elapses, the PCI server 48 assumes that the subscriber is out of radio coverage or has quit the PCI application.

For implicit (or automatic) deregistration, the message flow is the same as illustrated in FIG. 13. The PCI server 48 sends to the PDA 30 a deregistration request containing registration information about the subscriber. The PCI server 48 also sets a timer during which it will wait for a response from the PDA 30. When the PDA 30 receives the deregistration request, it responds with registration acknowledgement which contains the registration information currently known to PDA. When the PCI server 48 receives the registration acknowledgement, it updates the subscriber's registration status based on information in the acknowledgement. The PCI server 48 also updates the wireless messaging time-stamp associated with the subscriber to the current time. If the PCI server 48 does not receive an acknowledgement within a predetermined time as described above, the PCI server 48 assumes that the subscriber is no longer registered and removes all references to the subscriber from the service profile cache 51.

Sending and receiving e-mail wireless messages involves two types of message flows: sending messages from the PDA 30 to the PCI server 48 and from the PCI server 48 to the PDA 30.

FIG. 14 is an illustrative example of a message flow sending an e-mail from a PDA 30 to a PCI server 48. When a subscriber sends an e-mail notification from the PDA 30, the PDA 30 forwards the e-mail notification to the PCI server 48. The body of the e-mail contains, for example, "to;from;subject;cc" information (line 308). The PCI server acknowledges this notification (line 310). If the e-mail is longer than can be transmitted in a single message, the PDA 30 segments the e-mail into multiple, sequentially numbered packets and sends them to the PCI server (lines 312, 316, 320). Each packet sent from the PDA is responded to with an acknowledgement containing the reception status of the message and the sequence number it is acknowledging (lines 314, 318, 322). The PDA 30 and PCI server 48 use the sequence number to maintain a sequential flow of packets. Out of sequence packets are discarded. Once all of the packets are received, the PCI server 48 puts them into their original order using the sequence number and forwards the now assembled e-mail to a message transfer agent, which then forwards the e-mail to its intended destination.

The PDA 30 starts a timer each time it sends out an e-mail. If the PDA 30 does not receive an acknowledgement after a predetermined time (for example ten seconds), the send operation is aborted and the e-mail is stored in a local outbound queue for redelivery in the future.

When an e-mail is being delivered from an PCI server 48 to a PDA 30, a similar message flow is used. The only difference is that the PCI server 48 initiates the flow and sends the initial messages instead of the PDA 30.

Retrieving e-mail involves two types of message flows: retrieving undelivered e-mail addressed to the PDA 30 and retrieving e-mail delivered a messaging system, such as a wireline e-mail system. When a subscriber is out of radio coverage or is not registered with PCI, the PCI sends e-mails addressed to be delivered to the PDA (PDA-bound e-mail) to an external mail storage system. The PCI server will also send certain e-mail directly to an external mail storage system (MS-bound e-mail), such as the subscriber's wireline e-mail connected to his or her personal computer, according to the subscriber profile stored in the PCI database.

A registered subscriber can retrieve PDA 30 bound e-mail at any time by starting "FETCH" operation. The PCI will send the PDA bound mail from the external mail storage and will also summarize MS-bound e-mail.

An illustrative example of the message flow between the PDA and the PCI server for retrieving undelivered PDA bound e-mail is shown in FIGS. 15(a) and (b). If there are no MS-bound messages, an illustrative message flow is shown in FIG. 15(a). The PDA 30 sends a fetch request to the PCI server 48 (line 324) and starts a timer, which waits for an acknowledgement. If no acknowledgement is received within a predetermined time, for example twelve seconds, the PDA 30 assumes it is out of radio coverage and informs the subscriber to try again later. In response to the request, the PCI server 48 logs into an external mail storage system specified in the subscriber's profile. If any PDA-bound e-mail is stored in the external storage system, the PCI server 48 will (a) move the PDA bound e-mail from the external mail storage system into a pending to a in the PCI server; (b) send an acknowledgement to the PDA indicating the number of PDA bound e-mail now residing in the pending area; and (c) initiate delivery of these PDA bound e-mail from the pending area to the PDA (line 326).

If there are MS-bound e-mail messages, an illustrative message flow is shown in FIG. 15(b). The PDA sends a fetch request (line 328) and starts a timer. Whenever the PCI server sends a summary message, it starts a timer. If the PCI

server 48 does not receive an acknowledgement within a certain predetermined time, for example ten seconds, it will assume that the PDA 30 is out of radio coverage, abort the send operation, and discard the summary information. In response to the request, the PCI server 48 will (a) send an acknowledgement to the PDA indicating the number of MS-bound e-mail present (line 330); (b) extract summary information from those messages; and (c) send the summary to the subscriber's PDA (line 332). When the PDA receives an acknowledgement from the PCI server, it informs the subscriber based on the contents.

Summary information for the MS-bound e-mail is formatted into one ASCII text per e-mail and sent to the PDA. If the summary information, or the number of summarized e-mail require more than one message, the PCI server 48 splits the summary information into multiple, sequentially numbered segments and sends each segment in a separate message (lines 336, 340). Each message from the PCI server 48 is responded to by the PCI server with an acknowledgement containing the reception status of the message and the sequence number it is acknowledging (lines 334, 338, 342). Out of sequence messages are discarded. Once all of the packets are received, the PDA 30 puts them into their original order using the sequence number.

Once the summary information describing the MS-bound e-mail messages is reviewed, the subscriber may start a FETCH operation to retrieve these MS-bound e-mail messages. FIG. 16 is an illustrative example of a message flow between the PDA 30 and the PCI server 48 retrieving MS-bound e-mail. The subscriber selects an MS-bound e-mail message to be received. The PDA 30 sends a retrieve request to the PCI server 48 containing the message selected by the subscriber (line 344). The PCI server 48 responds with an acknowledgement (line 346). The PCI server 48 logs into the external message storing system specified in the subscriber's service profile and moves the MS-bound e-mail specified in the request out of the storage system into a pending area in the PCI server 48. The PCI server 48 initiates a send operation which delivers the e-mail in the same manner as discussed above.

Cross media notification (e.g., PDA notification of voice mail message receipt) is sent to the PDA 30 using the same delivery as a wireless e-mail message to the subscriber. The PCI server 48 originates the notification e-mail and the e-mail subject is "message notification". The body of the notification e-mail contains the message sender's address (i.e., the phone number for a voice mail), the date and time the message arrived at the PCI; the type of media, (i.e., voice mail, FAX, e-mail or other); whether the message is marked urgent (if detectable); the length of the message (for example, in minutes for a voice mail message); and, if appropriate, the subject of the message.

Profile management allows the subscriber to modify wireless messaging and CallCommand services by updating certain elements in the subscriber's service profile stored in the PCI database 44 and the service profile cache 51 in the PCI server 48. Profile information is not stored locally on a PDA 30. Updating the subscriber's profile using a PDA 30 always requires the subscriber to have a profile download from the PCI.

Profile management involves two types of message flows, profile download and profile upload. FIG. 17 is an illustrative example of the message flow between the PDA 30 and the PCI server 48 for a profile download. As indicated above, any profile change requires a profile download because the profile is never stored in the PDA 30. A subscriber starts a profile management application on a PDA 30 and requests a

profile download. The PDA 30 sends a download request to the PCI server and requests a copy of the subscriber's modifiable profile elements to be downloaded to the PDA 30 (line 360). The PCI validates the identity of the subscriber through its subscriber ID and password. If the subscriber's identity is not validated, the PCI sends an acknowledgement and an error code and terminates the profile update session. If the subscriber's identity is validated, the PCI downloads the subscriber's modifiable profile elements (lines 362, 366, 370). Attached as Appendix C is a list of tags for modifiable profile elements. The PDA 30 acknowledges the received data (lines 364, 368, 372). The PDA starts a timer after sending the download request. If the PDA does not receive an acknowledgement or data from the PCI server within a predetermined amount of time, for example, ten seconds, it assumes that it is out of radio coverage and informs the subscriber to try again later. The PCI server 48 starts a timer each time it sends out data to the PDA 30. If the PCI server 48 does not receive an acknowledgement from the PDA 30 within a predetermined time, for example ten seconds, it will abort the profile download operation.

Once the subscriber finishes editing the profile on the PDA, a profile upload request is issued. An illustrative example of the message flow between the PDA 30 and the PCI server 48 for a profile upload is shown in FIGS. 18(a) and (b). After the subscriber issues a profile upload request, the PDA 30 sends an upload request to the PCI server 48 requesting permission to send the updated profile elements (step 374). The PCI server 48 validates the identity of the subscriber, for example by checking the subscriber ID and password, and checks if there is an associated download request issued by the same subscriber. The check for an associated previous download request is necessary so that the PCI server 48 is sure that the profile the subscriber wants to change is the profile that the PCI server 48 has just sent. If the subscriber's identity is not validated, or there is no associated download request packet, the PCI server sends an error code to the PDA 30 and terminates the profile upload session. If the subscriber's identity is validated and there is an associated download request, the PCI server 48 honors the request by sending an acknowledgement and a status code of "OK" to the PDA 30 (line 376). When the PDA 30 receives the "OK", it formats the updated profile elements and sends them to the PCI server 48 in the same way the profile was sent to the PDA 30 during the download phase (lines 378-386). If no error is detected, the PCI server 48 sends the updated profile elements to the PCI database 44 to commit the change. After a confirmation is received from the PCI database 44, the PCI server 48 sends an acknowledgement with status code of "OK" to the PDA to confirm and conclude the profile upload session (line 388), as shown in FIG. 18(b).

FIG. 18(b) is an illustrative message flow when the PCI server 48 detects errors in an uploaded profile. The upload proceeds as above (lines 390-398). If the PCI server 48 detects errors in the updated profile elements it responds with an error message to notify the subscriber about the invalid profile element (line 400). The PDA acknowledges receipt of the error message (line 402). The PCI server 48 sends the invalid profile elements in a similar way as the profile was sent to the PDA 30 during the download phase (lines 404, 406).

The PDA 30 starts a timer when it sends out an upload request or sends out data. If the PDA 30 does not receive an acknowledgement from the PCI server 48 within a certain predetermined time, it will abort the profile upload operation and inform the subscriber to retry at a later time.

V. Wireless E-mail Messaging Services

PCI includes several wireless text message sending, receiving, and service control features. PCI's wireless text messaging services are based on three network-based capabilities:

message integration combining voice message notification, voice mail, telephone calls, e-mail, and fax;

message routing and delivery, i.e., the PCI is a wireless and wireline network gateway;

database access, i.e., subscriber profile, authentication, and validation.

The PCI uses personal communications service integration capabilities to integrate the wireless service capabilities available to the subscriber. This is accomplished by providing the subscriber with control over the message routing and delivery by the subscriber accessible "subscriber profile" stored in the PCI. The subscriber profile contains subscriber programmed instructions on message receipt, origination, and notification. Thus, PCI operates as a messaging gateway for providing access to multiple wireline and wireless networks, while using subscriber profile information to control sending and receiving options. PCI allows wireless service providers to integrate the voice messaging, e-mail, and fax message services for one subscriber through a single telephone number. Thus, one phone number may provide a single link between the service provider and the subscriber's voice and data communications lines.

The message sending features include communications across disparate networks and broadcast communications. A subscriber may send voice mail, e-mail, and fax messages between different service providers and networks. A subscriber may also send broadcast e-mail and fax messages, which broadcasts may mix e-mail and fax messages within a single distribution list. For example, the subscriber may type a message on a PDA and send it to a distribution list over a wireless network. The distribution list may, for example, direct the PCI to deliver the message to the office as an e-mail and to a client as a fax.

The message receiving features include personal number addressing, selection of message receipt media format, selection of cross-media message notification, and selection of message screening and delivery options. A subscriber may receive voice (e.g., phone), voice mail notification, e-mail, and fax communications under a single personal telephone number. A subscriber may direct e-mail and fax delivery based on selected parameters, such as time-of-day, day-of-week, etc. A subscriber's media message notification, voice mail notification of e-mail or fax messages, e-mail notification of voice mail or fax messages, and fax notification of e-mail or voice mail messages may be delivered to the subscriber based on selected options and parameters.

Alternatively, if the subscriber's wireless terminal is not activated, e-mail messages may be automatically routed to alternate destinations as defined by the subscriber's profile. For example, the subscriber may not want to receive all telephone calls at a visiting location to avoid unnecessary interruptions and unwanted incoming call charges. The subscriber directs the PCI to send notification of phone calls to the pager and to route the call to voice mail. Once notified, the user can determine from the phone number included in the pager notification whether to call the person directly, check voice mail, or ignore the call until a later time. The subscriber may also direct which messages are to be routed to the subscriber's current serving network, which are to be sent to another network, and what media is to be used to receive certain messages. The subscriber may also

designate, for example, that if the wireless terminal is off, all text messages to be sent to e-mail and all voice messages are to be sent to voice mail.

The PCI service control features include supporting subscriber profile management, supporting personal mobility across wireless and wireline networks, and supporting wireless terminal mobility. A subscriber's profile may be updated by sending text messages from a PDA over a wireless network or DTMP (touch-tone) messages from either a wireline or wireless terminal. The subscriber may program the profile to select media for receiving and sending information; select cross media for message notification; select message screening and delivery options; select single voice mailbox storage (for subscriber's with more than one voice mailbox); and select a PCI service password. All of these options may be maintained over wireless or wireline terminals. The subscriber may automatically register and deregister a wireless terminal thus updating the subscriber's profile to receive or reroute messages as preprogrammed in the profile.

The wireless data network provides data transport between the PCI server 48 and the subscriber using a wireless data terminal, such as a PDA 48. The wireless data network may connect to the PCI server in a variety of ways, using a variety of protocols. For example, the wireless data network may connect to the PCI using a leased line and run a proprietary protocol to connect the PCI server via standardized protocols such as TCP/IP.

Text messaging systems may be connected to the PCI server through for example, Frame Relay, SMDS, ISDN, leased line interface, or other transport mechanism effective for supporting data communications may be used. An inter-message handling system protocol, such as X.400 (in which case X.400 gateway conversion is needed), or Internet SMTP or other protocols supported by an interworking unit terminating the data transport interface, may be used to forward messages between the PCI server 48 and the system accessing the PCI.

The PCI server will preferably support sending and receiving faxes in the T.434 format. The PCI server may also preferably support sending and receiving faxes using the simple mail transfer protocol (SMTP) supported by the TCP/IP transport protocol.

FIG. 19 shows an illustrative embodiment of a PCI service which supports text messaging systems. In this example, a subscriber has a personal computer 402 at the office connected to a local area network (LAN) 414 and an enterprise text messaging system (for example, a local network e-mail) 413, a personal computer at home 416, and a wireless terminal, such as PDA 30 that may send and receive messages. All of these devices are connected to the PCI. For example, the subscriber's home personal computer 416 may be connected to the PCI 40 via a modem and a wireline data network 418 over either a PSTN or ISDN.

Persons connected to the LAN may send text messages to the subscriber by using the local text messaging system without using the PCI. That is, the user of computer 420 can send an e-mail to the subscriber's office computer 412 without entering the PCI node 40. Because the enterprise text messaging system 413 is connected to PCI, all enterprise messaging users may send messages to and receive messages from all PCI subscribers (including those not connected to the local text messaging system 413) by using an appropriate PCI address.

A person connected to a different enterprise messaging system, such as a second text message handling system 422, can send messages to the subscriber on the first message handling system 413 by routing the message through the PCI Server 48.

PCI subscribers are assigned a single personal telephone for both voice and data communication. For example, an E.164 address (i.e., a telephone number) may be assigned to a PCI subscriber to use as the single PCI address. These phone numbers may be geographically based according to current PSTN architecture, but it is also possible to use portable universal numbers. Fifteen digit number formats may be desirable to permit sub-addressing. For example, a message destined for a PCI subscriber may be addressed to the subscriber's telephone number, e.g., 201-555-5555. If an originating mail system such as a LAN mail system or third party message handling system requires a domain identifier, the originator may have to specify 201-555-5555 @ PCI, or on the Internet 201-555-5555 @ pci.net. When the PCI server 48 receives the message, it will look at the subscriber's profile stored in the call process request database 222 stored in a PCI database 44 to determine how to handle the incoming message. An example of a few of the options that PCI may provide for the subscriber are to:

- send the message to the subscriber's wireless PDA;
- send the message to the subscriber's wireline computer at home;
- send the message to the destination text messaging system at the office;
- send a notification of an incoming message to the wireless data terminal and the actual message to the text messaging system;
- send the message to any or all of the above;

The subscriber may send text messages over the wireless data network or wireline data network to the PCI server 48. The PCI server 48 consults with the subscriber's profile at the PCI database 44 and forwards the message to the appropriate destination, depending on the routing destination found in the profile. Text messaging systems not connected to the PCI 40 may send text messages to PCI subscribers by using another network connected between the sender's text messaging system and the PCI subscriber's text messaging system. For example, the non-connected text message may be connected to a PCI over the Internet.

The flow for wireless messaging is now described.

The flow for a PCI subscriber receiving an e-mail message to a wireless PDA 30, for example, is as follows. An e-mail message is sent from a wireline or wireless sender to a PCI subscriber and arrives at the PCI server 48. The incoming e-mail contains a recipient address in the format of "201-555-5555 @ pci.net" where 201-555-5555 is the subscriber's ten-digit personal number and pci.net is the PCI server's domain name in the Internet.

The PCI server 48 checks the subscriber's service profile, either from the profile service cache 51 in the PCI server or by downloading the subscriber profile from the PCI database 44 into the cache 51 to determine how to process the e-mail message. The profile contains screening and routing information and cross media notification information. The PCI server 48 uses this information to send incoming email to an actual destination address that can be a wireless, wireline, or paging address using, for instance, the UDP/IP protocol over a wireless data network; the Internet SMTP protocol over the Internet wireline network; or the Telocator Alpha Numeric Protocol (TAP), respectively. In this case, the subscriber has programmed into the subscriber profile to have the e-mail sent to a PDA 30. The PCI server 48 receives the e-mail message and forwards it to the wireless data network programmed into the profile. The e-mail is transmitted over a wireless data network 39 for receipt by the PDA 30.

If the e-mail cannot be delivered, the PCI server returns the e-mail to the original sender with a short description of why the delivery was unsuccessful, using the SMTP protocol.

If an e-mail message is to be delivered to an alphanumeric paging address, the PCI server translates the e-mail message into a paging message and sends the paging message to the paging network specified in the subscriber profile. The protocol between the PCI server and the paging network is the Telocator Alphanumeric Protocol (TAP). The PCI server formats the paging message into a maximum page limit with a maximum number of characters per page. For example, the page limit may be two pages and a maximum of 256 characters per page. The PCI server does not verify whether a paging message is actually delivered by the paging service provider. It will, however, verify that the message was successfully sent to the paging service provider. Because the PCI server does not provide this verification, it is assumed that messages sent to a pager arrive successfully at the pager.

If the subscriber profile contains an option for voice message notification of e-mail messages, the PCI server generates and sends a digitized prerecorded voice announcement to the address specified in the subscriber service profile. The protocol used to deliver the voice message notification is the AMIS-Analog Protocol.

In this illustrative embodiment, a preferred PCI server node functions as an X.400 message transport agent or SMTP router and routes messages destined for PCI subscribers and to those destined for users connected on other systems. In the case of an X.400 message transfer agent (MTA), X.400 addresses are used to internally represent subscriber addresses. The translation from the "user friendly" subscriber addresses such as E.164 numbering to the X.400 address would be done via a look-up table (ROM or other memory device) at the PCI access module or the X.400 gateway. Destination or source addresses from users on other networks are not converted to X.400 addresses, but are left in the native address format of the sending or receiving system. An X.400 gateway address may be added to the message header, however, to allow PCI to route the message to an appropriate gateway.

The PCI server 48 is responsible for delivering a message to the subscriber listed in the destination field of the message. In a simple case, the subscriber has an X.400 or Internet mailbox accessible to the PCI via one of its access connections. Alternatively, the subscriber profile may contain forwarding addresses which route the message for delivery to unusual destinations. For example, the subscriber's mailbox may reside on another message handling system, a wireless data network, wireline data network, or PSTN destination associated with a fax machine. The delivery of such a message to a final destination is handled by an interworking unit which is responsible for doing address translation and, if necessary, format translation as defined by the subscriber profile entry.

For subject e-mail screening, the subject field is analyzed to determine if a match exists before comparing the address field. If the subject field matches an entry on the screening list, the treatment for a matched entry will occur. That means, in this illustrative embodiment, that subject screening takes precedence over address sender screening. That is, if e-mail originated from an address that is excluded from the e-mail screening address list, the e-mail will still be delivered according to the screening criteria.

If the PDA 30 is not registered for the wireless messaging service or if the PDA 30 is out of radio coverage at the time the message arrives at the PCI server 48, the message will be sent to the subscriber's external message storage system, such as the text message system 413.

VII. Message Flows

PCI wireless messaging involves three types of message flow. The first is sending a message from one subscriber to

another, the second is receiving a message regardless of whether the subscriber is using a wireless or wireline terminal, and the third is sending a message to a non-subscriber.

FIG. 20 is an illustrative example of the message flow of a PCI wireless subscriber sending a message. The PCI user submits a message 502. The message is received by a message transfer agent in the PCI server. The MTA copies and temporarily stores the originating and destination addresses 504. The MTA sends to the mobility manager function in the PCI server a request to validate the sending user as a PCI subscriber 506. The mobility manager sends this validation request to the PCI database and waits for a response 508. Upon receipt of an affirmative validation from the PCI database, the mobility manager sends the validation response to the MTA 510, 512. The MTA then sends the mobility manager a request for the address of the user's home MTA 514. The mobility manager routes this request to the PCI database 516. Upon receipt of a response from the PCI database, the mobility manager routes the home MTA address to the MTA 518, 520. The MTA then routes the message to the home MTA 522. If a third-party PCI database must be consulted, the home MTA request will be directed from the PCI database to a third party PCI database 524, 526.

FIG. 21 illustrates an example of the message flow of a wireless PCI user receiving a message. When the PCI receives a message from a subscriber, the MTA in the PCI server copies and temporarily stores the destination address and the message 530. The MTA sends to the mobility manager function in the PCI server a request for the PCI subscriber's user profile 532. The mobility manager will retrieve this profile request from the PCI database 534. (If a third party PCI database is involved, the local PCI database contacts the third party PCI database through a switch transfer point 536, 538.) Upon receipt of the subscriber's profile from the PCI database 540, the mobility manager requests the message from the MTA using a "message forward request" message 542. When the mobility manager receives the message from the MTA 544, the mobility manager processes the message as indicated by the subscriber's profile, which may involve media conversion or screening 546. After processing the message, the mobility manager sends the message to the MTA for delivery 548, 550. Alternatively, the PCI server mobility manager function may directly deliver the message to the termination receiver 552.

FIG. 22 illustrates an example of a message flow from a PCI wireless subscriber to a non-subscriber. When the MTA receives a message from a PCI subscriber 560, the MTA copies and temporarily stores the originating addresses and the message 562. The MTA sends the mobility manager a request to validate the originating address as a PCI subscriber 564. The mobility manager will send this validation request to the PCI database and wait for a response 566. When the mobility manager receives an affirmative validation response from the PCI database 568, the mobility manager sends the validation response to the MTA 570. Next, the mobility manager sends to the PCI database a request for the PCI subscriber's profile 572. Upon receipt of the subscriber's profile from the PCI database 574, the mobility manager requests the message from the MTA using a "message forward request" 576. Upon receipt of the message from the MTA 578, the mobility manager processes the message as indicated by the user's profile, which may require media conversion or obtaining the addresses for the distribution list for the message 580. After processing the message, the mobility manager sends the message to the

MTA for delivery 582, 584. Alternatively, the MTA may directly deliver the message 586.

VIII. The PDA Application

To better understand the capabilities of PCI and PDA/PCI server interface, a discussion of the PDA user interface is helpful. The user interface is application software residing in the PDA. This software is described by describing the screens displayed on a PCI subscriber's PDA screen. The following discussion is for an illustrative embodiment of the PDA user interface. A person skilled in the art recognizes that the interface may be implemented in a myriad of ways.

FIG. 23 is an illustrative example of a PDA user interface main menu. The menu allows the user to enter the CallCommand or wireless messaging services, update the user profile, or check the status of the system by clicking on buttons 610, 612, 614, 616, respectively.

FIG. 24 shows a computer screen after "status request" 616 is selected. The status request screen shows that there are five local originating messages (waiting to be sent by the PDA) and three outgoing messages (waiting to be retrieved) in boxes 618, 620, respectively. The various services' status is also displayed. As seen in FIG. 29, this subscriber's wireline registration is on, as seen in box 622. This registers the subscriber on a particular wireline telephone, seen in box 624. This registration will direct calls to this phone number. The status request also advises this subscriber about the status of the CallCommand and wireless messaging features, as seen in boxes 626, 628.

FIG. 25 is an illustrative example of a screen if the subscriber selected "Wireless Messaging" 512 on the main menu (FIG. 23). The subscriber will be connected to the wireless messaging service if "YES" 642 is clicked.

FIG. 26 is an example of a screen which may appear if the subscriber selected "Profile" 614 from the main menu (FIG. 23). If the subscriber selects "Fax" 644 from this screen, a screen such as that shown in FIG. 27 may appear, which allows the subscriber to enter a phone number into box 646 to which faxes will be directed. Turning on e-mail screening activates both the subject and address screening. Subject screening takes priority over address screening parameters.

If the subscriber selected "e-mail" 648 on the screen FIG. 26, a screen such as in FIG. 28 appears. The subscriber can select where e-mail messages should be delivered (destination screening) 650, where notification of e-mail receipt should be delivered (notification screening) 652, whether messages should be screened at all 654 and, if so, how they should be screened 656, 658.

The destination 650 allows the subscriber to select destinations for incoming e-mail. Messages that satisfy the screening requirement may be sent to two destinations (match A, match B). As shown in this illustrative example, e-mail received which match the subscriber's preprogrammed screening criteria are to be delivered only to a wireline e-mail, such as the subscriber's personal computer at the office, because match A 660 and match B 662 designate the same destination. All received e-mail messages which do not meet either criteria ("not matched") are sent to a selected fax machine 664, for example, the fax machine at the subscriber's office.

The subscriber also indicates where notification of a received e-mail should be sent 652. Notification for all e-mail messages meeting the screening requirement should be sent to a selected fax machine 666. The PCI network will select information about the e-mail origination such as the author, recipient, and subject matter and convert it to a facsimile format and send the message to a fax machine. Notification of all e-mail that does not meet the screening

criteria are sent to a pager 668. The PCI network will take the originating message information and turn it into alphanumeric information according to the TAP protocol and send it to the subscriber's pager. If the screening option is turned off, notification of all incoming e-mail is sent to voicemail 670. The PCI network will convert the origination information from text to synthesized speech and send the information to a selected voice mailbox.

The user may also select whether to screen incoming e-mail messages at all 654. If the screening is on, the user may screen e-mail based on the originating address 656 or subject matter 658.

FIG. 29 is an illustrative screen which the subscriber may use to edit e-mail screening according to address by clicking box 656 (FIG. 28). The subscriber may input new e-mail addresses into box 672 and add them to a list by clicking a box 674 or select addresses already entered to be included in a screening criteria as seen in box 676. For example, the user may want e-mail messages originating from the following addresses to be routed according to the screening criteria: cclstanp, ccltrizo, and cclrupin. E-mail messages originating from these addresses will be routed and notified according to the criteria selected on the screen illustrated in FIG. 28.

If the user selected to edit the "subject" a screening criteria based on "subjects" by clicking box 658 (FIG. 28), a screen such as that illustrated in FIG. 30 is presented. The user may type in to boxes 678 particular subjects which should be routed according to a screening criteria. The subject will search the incoming e-mail origination information to determine the subject of the e-mail. Subjects may include "urgent", "personal", the name of a client or project, etc.

IX. Billing

Billing operations is supported by an Automatic Message Accounting Network Function. The automatic network accounting measures, collects, formats and outputs network usage information to upstream billing and other operation application and service purposes. Preferably, automatic message accounting data is collected at various stages of service flows across network equipment and services.

X. Conclusion

An improved e-mail system is described which provides enhanced delivery and notification options and profile update capabilities.

While the invention has been described by the reference to specific embodiments, this was for purposes of illustration only and should not be construed to limit the spirit or the scope of the invention.

APPENDIX 1

Glossary of Acronyms

AIN	advanced intelligent network
ANI	automatic number identification
APS	alphanumeric paging server
ATM	asynchronous transfer mode
CC	CallCommand
CCDB	call contact database
CO	central office
CPE	consumer premises equipment
CFR	call process request
D&R	data and report database
DRS	data and report subdivision
DTMF	dual tone multiple frequency
GDI	generic data interface
HLR	home location register
IP	intelligent peripheral

APPENDIX 1-continued

Glossary of Acronyms

IPi	intelligent peripheral interface
ISDN	integrated signaling digital networks
LEC	local exchange carrier
MOC	maintenance and operation console
MSAP	multiple services application platform
MTA	message transfer agent
PCI	personal communications internetwork
PDA	personal digital assistant
POP	post office protocol
PSN	public packet switched network
PTN	public switched telephone network
SDMS	switched multimegabit digital service
SMS	service management system
SNI	service network interface
SPACE ®	service provisioning and creation environment
SPC	service profile cache
SSP AT	service switching point access-tandem
TAP	teletext alphanumeric protocol
TCAP	transaction capable application program
TCP	transaction capable processing
VLR	visiting location register

APPENDIX 2

PCI Profile Element	GDI Tag ID	Type and Max Length
Personal Data (1)		
Subscriber ID Number	1	d10
Password	2	c8
E-mail Address	3	c32*
V-Mail System Phone Number	4	d10
V-Mail System Mailbox Number	5	d10
Paging Terminal Phone Number	6	d10
Paging PIN	7	c8
Mail Storage Account ID	8	c32*
Mail Storage Password	9	c8
Wireless Provider Subscriber ID	10	c10
Wireless Voice Phone Number	11	d10
Fax Number	12	d10
Profile Version Number	13	n4
Call Command Service Profile (2)		
Call Command Registration Status	21	c1
Wireless Registration Number	22	d10
E-Mail Routing (3)		
Primary Destination 1	31	c1
Primary Destination 2	32	c1
Primary Notification 1	33	c1
Secondary Destination	34	c1
Secondary Notification	35	c1
Default Destination	36	c1
Default Notification	37	c1
E-Mail Subject Screening (4)		
E-Mail Screening Status	41	c1
Subject1	42	c1
Subject2	43	c1
Subject3	44	c1
Subject4	45	c1
Subject5	46	c1
E-Mail From Screening (5)		
E-Mail Screening Status	41	c1
From1	52	c12
From2	53	c12
From3	54	c12
From4	55	c12
From5	56	c12
From6	57	c12
From7	58	c12
From8	59	c12

APPENDIX 2-continued

From9	60	c12
From10	61	c12
From11	62	c12
From12	63	c12
From13	64	c12
From14	65	c12
From15	66	c12
Voice Mail Profile (6)		
V-Mail Screening Status	71	c1
From1	72	c12
From2	73	c12
From3	74	c12
From4	75	c12
From5	76	c12
Primary Destination	77	c1
Primary Notification 1	78	c1
Secondary Destination	79	c1
Secondary Notification	80	c1
Default Destination	81	c1
Default Notification	82	c1

APPENDIX 3

Tag	Description
001	Password for subscriber registration
002	Password for retrieval of messages from message store
003	FAX Machine Number
004	Call Command Status flag (1 = ON and 0 = OFF)
005	Wireline Registration Number for remote call forwarding
006	E-mail screening status flag (1 = ON and 0 = OFF)
007-021	E-mail "From" screening 1 through 15
022-026	E-mail "Subject" screening 1 through 5
027	E-mail Destination A - Screening Criteria Match
028	E-mail Destination B - Screening Criteria Match
029	E-mail Notification - Screening Criteria Match
030	E-mail Destination - Screening Criteria Not Match
031	E-mail Notification - Screening Criteria Not Match
032	E-mail Notification - Screening OFF
033	Voice-mail Screening Status Flag (1 = ON and 0 = OFF)
034-038	Voice-mail "From" Screening 1 through 5
039	Voice-mail "Urgent" Flag (1 = ON and 0 = OFF)
040	Voice-mail Notification - Screening Criteria Match
041	Voice-mail Notification - Screening Criteria Not Match
042	Voice-mail Notification - Screening OFF

We claim:

1. A communication system comprising

a plurality of wireless subscribers each having a unique address to which electronic mail messages and fax, pages, and voice communications may be sent;

means for storing a profile for each wireless subscriber, said profile containing for each of said wireless subscribers for (1) screening information for selectively screening incoming messages based on at least one of message origin, time of day, and day of week; (2) routing information for transmitting the message to the subscriber or for returning the electronic message to the sender with a short description of why delivery was unsuccessful; (3) and cross-media notification information for translating the electronic message into a paging message to be sent to a paging network and for sending a voice message notification of the electronic mail message; and

means for receiving an electronic mail message addressed to one of said subscribers and responsive to the infor-

mation in said profile for said one subscriber for causing one of (1) delivering the electronic mail message to said one subscriber; (2) returning the electronic mail message to its sender with a short description of why delivery was unsuccessful if the electronic mail message cannot be delivered to said one subscriber; (3) translating the electronic mail message to a paging message and delivering the paging message to a paging network if the electronic mail message is to be delivered to a paging address; and (4) sending a voice announcement to the address specified in said one subscriber's profile.

2. A method for receiving an electronic mail message from a sender addressed to a wireless terminal at the same telephone number to which fax, pages, and voice communications may be sent, said method comprising the steps of:

determining from a pre-stored profile for the user how to process the electronic mail message, the profile containing (1) screening information for selectively screening incoming messages based on at least one of message origin, time of day, and day of week; (2) routing information for transmitting the message to the user or for returning the electronic message to the sender with a short description of why delivery was unsuccessful; (3) and cross-media notification information for translating the electronic mail message into a paging message to be sent to a paging network and for sending a voice message notification of the electronic mail message;

and, responsive to said determining step, causing the message to be delivered to a forwarding address different from the address of the user's wireless terminal.

3. The method for receiving an electronic mail message from a sender addressed to a wireless terminal at the same telephone number to which fax, pages, and voice communications may be sent, said method comprising the steps of:

determining from a pre-stored profile for the user how to process the electronic mail message, the profile containing (1) screening information for selectively screening incoming messages based on at least one of message origin, time of day, and day of week; (2) routing information for transmitting the message to the user or for returning the electronic message to the sender with a short description of why delivery was unsuccessful; (3) and cross-media notification information for translating the electronic mail message into a paging message to be sent to a paging network and for sending a voice message notification of the electronic message;

said determining step further comprising consulting a distribution list containing a plurality of delivery destinations, each destination having a delivery format of one of a facsimile and an electronic mail format, and responsive to said determining step causing the message to be delivered to the plurality of destinations from the distribution list, the message being routed to each destination in the delivery format prescribed by the distribution list.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 5,742,668
DATED : April 21, 1998
INVENTOR(S) : D.M. Pepe, et al

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the title page, item [54] and Col. 1, line 1, "Massaging" should read
-- Messaging --.

Signed and Sealed this
Twenty-third Day of June, 1998

Attest:



BRUCE LEHMAN

Attesting Officer

Commissioner of Patents and Trademarks

EXHIBIT G

[54] DATABASE USAGE METERING AND PROTECTION SYSTEM AND METHOD

[75] Inventor: Victor H. Shear, Bethesda, Md.

[73] Assignee: Electronic Publishing Resources, Inc., Bethesda, Md.

[*] Notice: The portion of the term of this patent subsequent to May 2, 2006 has been disclaimed.

[21] Appl. No.: 310,938

[22] Filed: Feb. 16, 1989

Related U.S. Application Data

[62] Division of Ser. No. 918,109, Oct. 14, 1986, Pat. No. 4,827,508.

[51] Int. Cl.³ H04K 1/00

[52] U.S. Cl. 380/4; 380/25

[58] Field of Search 380/4, 3, 16, 25

[56] References Cited

U.S. PATENT DOCUMENTS

- 3,609,697 9/1971 Blevins .
- 4,120,030 10/1978 Johnstone .
- 4,168,396 9/1979 Best .
- 4,200,913 4/1980 Kuhar et al. .
- 4,232,193 11/1980 Gerard .
- 4,236,217 11/1980 Kennedy .
- 4,253,157 2/1981 Kirschner et al. .
- 4,305,131 12/1981 Best .
- 4,306,289 12/1981 Lumley .
- 4,319,079 3/1982 Best .
- 4,361,877 11/1982 Dyer et al. .
- 4,422,486 4/1984 Mayer .
- 4,433,207 2/1984 Best .
- 4,434,464 2/1984 Suzuki et al. .
- 4,446,519 5/1984 Thomas .
- 4,454,594 6/1984 Heffron et al. .
- 4,458,315 7/1984 Uchenick .
- 4,462,076 7/1984 Smith, III .
- 4,462,078 7/1984 Ross .
- 4,471,163 9/1984 Donald et al. .
- 4,494,156 1/1985 Kadison et al. .
- 4,513,174 4/1985 Herman .
- 4,528,588 7/1985 Lofberg .
- 4,553,252 11/1985 Egendorf .

- 4,562,306 12/1985 Chou et al. .
- 4,562,495 12/1985 Bond et al. .
- 4,577,289 3/1986 Comerford et al. .
- 4,584,641 4/1986 Guglielmino .
- 4,588,991 5/1986 Atalla .
- 4,589,064 5/1986 Chiba et al. .
- 4,593,353 6/1986 Pickholtz .
- 4,593,376 6/1986 Volk .
- 4,595,950 6/1986 Lofberg .
- 4,597,058 6/1986 Izumi et al. .
- 4,634,807 1/1987 Chorley et al. .

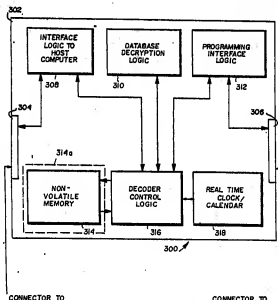
(List continued on next page.)

Primary Examiner—Salvatore Cangialosi
Attorney, Agent, or Firm—Nixon & Vanderhye

[57] ABSTRACT

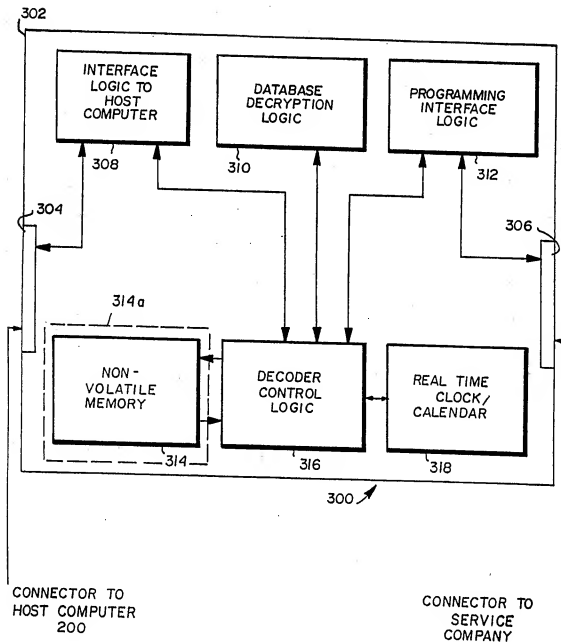
A "return on investment" digital database usage metering, billing, and security system includes a hardware device which is plugged into a computer system bus (or into a serial or other functionally adequate connector) and a software program system resident in the hardware device. One or more databases are encrypted and stored on a non-volatile mass storage device (e.g., an optical disk). A tamper-proof decrypting device and associated controller decrypts selected portions of the stored database and measures the quantity of information which is decrypted. This measured quantity information is communicated to a remote centralized billing facility and used to charge the user a fee based on database usage. A system may include a "self-destruct" feature which disables system operation upon occurrence of a predetermined event unless the user implements an "antidote"—instructions for implementing the antidote being given to him by the database owner only if the user pays his bills. Absolute database security and billing based on database usage are thus provided in a system environment wherein all database access tasks are performed at the user's site. Moreover, a free market competitive environment is supported because literary property royalties can be calculated based on actual data use.

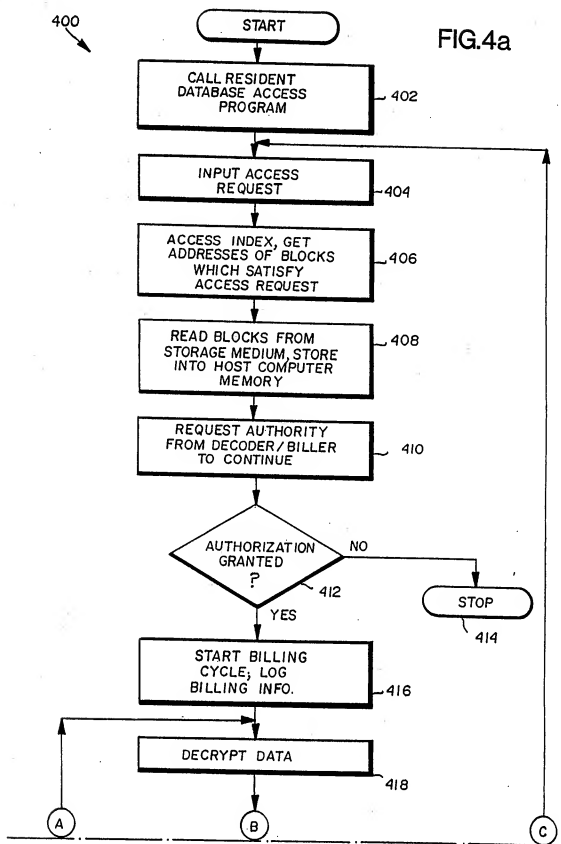
57 Claims, 5 Drawing Sheets



U.S. PATENT DOCUMENTS

4,644,493	2/1987	Chandra et al. .	4,688,169	8/1987	Joshi .
4,646,234	2/1987	Tolman et al. .	4,691,350	9/1987	Kleijne et al. .
4,652,990	3/1987	Pailen et al. .	4,696,034	9/1987	Wiedemer .
4,658,093	4/1987	Hellman .	4,701,846	10/1987	Ikeda et al. .
4,670,857	6/1987	Rackman .	4,713,753	12/1987	Boebert et al. .
4,680,731	7/1987	Izumi et al. .	4,740,890	4/1988	William .
4,683,553	7/1987	Mollier .	4,747,139	5/1988	Taaffe .
4,685,056	8/1987	Barnsdale et al. .	4,757,533	7/1988	Allen et al. 380/25
			4,791,565	12/1988	Dunham et al. 380/4 X
			4,827,508	5/1989	Shear 380/4





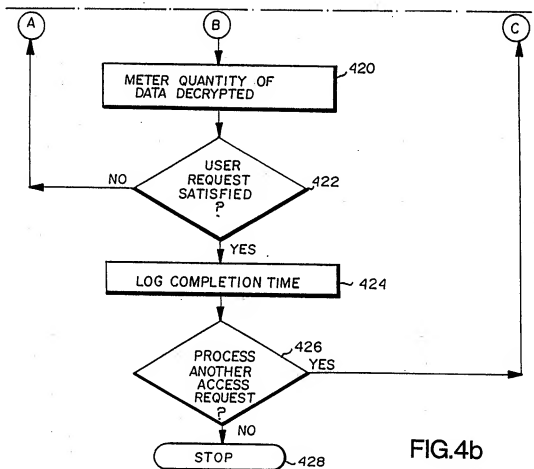


FIG. 4b

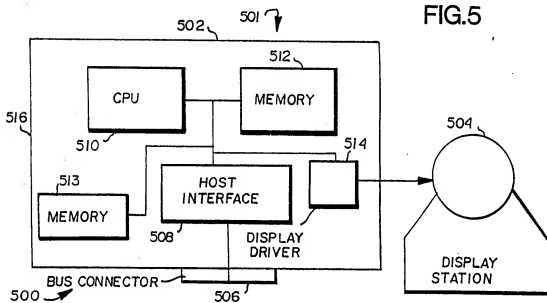


FIG. 5

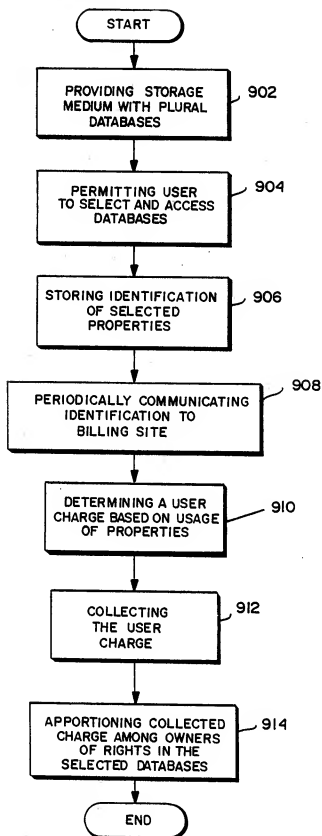


FIG.6

DATABASE USAGE METERING AND PROTECTION SYSTEM AND METHOD

This is a division of application Ser. No. 06/918,109, filed Oct. 14, 1986, now U.S. Pat. No. 4,827,508.

The present invention relates to regulating usage of a computer database. More particularly, the invention relates to techniques for preventing unauthorized use of an electronic digital information database and for measuring the utilization of the database by authorized users.

Information conveyed in electronic form is rapidly becoming the most valuable of commodities. Electronic digital databases now exist for a variety of different applications and fields of endeavor, and many businesses presently rely heavily on their ability to access those databases.

The value of being able to instantaneously, electronically access important, accurate information cannot be overestimated. Many of our daily activities depend on our ability to obtain pertinent information in a timely fashion. While printed publications and electronic mass media together fulfill most of the average person's informational needs and most often are the only source for full-text reference information, just about any effort to access information can benefit from the vast information handling capabilities of the computer. In today's fast-paced world, we quickly come to insist on and rely upon the most thorough and up-to-the-minute information available — often made possible only by electronic data processing and informational management technology. On-line, public databases, now a two billion dollar a year industry, are a case in point.

As the "information explosion" continues its course, more and more people will become dependent on electronically-stored information and people will continue to be willing to pay premium prices (when necessary) for access to and use of such information because of its usefulness and value to them. Currently, the principal resource for large, electronic information data bases are on-line (public) data base services such as Dialog Information Services, Mead Data Central, Dow Jones Information Services, Source, CompuServe, and many others. Most on-line data bases are abstract and/or bibliographic in content, and many are used primarily to access the document locations of specified information, rather than for the recall of the original document full-text.

Historically, personal computers have been used primarily for word-processing, modeling, and, to a lesser extent, the structured data base management of records. Technology that enables the user of, for example, a personal computer to search for, locate, and retrieve typically related full-text information from vast full-text data bases would be extremely useful and valuable.

The only viable way to make some kinds of information (e.g., information which must be constantly updated) available is to maintain centralized databases and permit users to access the centralized databases through telephone lines or other communication means. Until very recently, this method has been the most cost-effective way to offer access to electronic databases. Access to a centralized database can be controlled relatively easily, and users can be charged for using a centralized database in accordance with parameters which are relatively easy to measure (i.e., the amount of time the user is connected to the database computer, the number and

type of tasks the user requests, etc.). Moreover, because the database never leaves the central computer (each user is typically given access to only small portions of the database at a time), there is no danger of someone making unauthorized copies of the database.

However, centralized databases have important disadvantages. For example, it takes a relatively long time to manipulate information in a centralized database due to the relatively slow data transmission rates of standard communications channels and because the centralized database computer typically shares its resources among hundred or thousands of users at once. This can be a serious drawback if the user wishes to access a large volume of information or wishes to perform particularly complex data manipulation tasks. Also, it may take a long time during periods of peak database usage before communication can be successfully established with a centralized database computer, decreasing the utilization of the database and causing some users to become frustrated. Further disadvantages include the expense of establishing long-distance communications paths (e.g., WATS telephone line maintenance charges, long-distance direct-dial telephone charges, satellite channel costs, etc.) between distant user terminals and the central database computer, and the reliability problems associated with such communications paths. Moreover, the centralized computer facility needed to handle the access requests of many distant users simultaneously is extremely expensive to purchase and maintain.

With the advent of cheaper computer hardware and new, high density information storage devices (such as the optical disk and the bubble memory), it has become practical to give users their own copies of large and complex databases and permit users to access and manipulate the databases using their own computer equipment. Optical disks are capable of storing vast amounts of information at relatively low cost, are small enough to be sent through the mails, and can provide data at extremely rapid rates. Bubble memory devices provide some similar capabilities.

CD and related digital disk drives can currently store up to 225,000 pages of full-text information per removable diskette and can inexpensively maintain in excess of 1,800,000 pages of text simultaneously on-line. These technologies are ideal for personal computer information base libraries. CD drives use removable compact disks (essentially identical to an audio compact disk) the very low cost and enormous storage capacity has been predicted to result in an installed base of as large as one million drives to 10 million drives (including non-CD but related optical storage technology) by the end of 1990. Owners of "CD-ROM" and related drives will create an enormous demand for both lexical software and electronically published information base products. Mitsubishi Research Institute of Japan, for example, estimates that between 8,000 and 12,000 different CD-ROM publication titles will be on the market by the end of 1990.

Hence, it is now possible to store some databases on transportable, high-density information storage devices, and simply mail each user his own copy of the databases. The user can in this way be given exclusive access, via his own computer system, to local, on-site databases. Rapid access time is provided because access to the databases is exclusive rather than shared, and because data can be read from the database storage device by local high-speed I/O devices and transmitted over local high-speed I/O channels or networks. The

stored databases can be updated periodically if necessary by sending the user storage devices containing a new version of (or new portions of) the databases.

It is very expensive to build a database. One way to recover the costs of constructing and maintaining a database ("Return On Investment", or ROI) is to charge a flat subscription or access fee to each user subscribing to use the database. If this is the only billing method used, however, infrequent users of the database may be discouraged from subscribing, because they would be asked to pay the same cost a frequent user pays. Thus, many database owners charge subscribers a nominal subscription fee, and then periodically (e.g., monthly) charge users a fee calculated in accordance with the amount the user has used the database.

While it is easy to measure the amount someone uses a centralized database (e.g., simply time each access session length and store the time information with user identification information), there is no convenient way to measure the usage of a database residing on a user's own computer, or to convey such usage information to the owner of the database. Techniques are known for automatically, electronically measuring consumption of a commodity such as electricity, water or gas, storing the measurements in a memory device, and periodically downloading the stored measurements over a telephone line to a central billing computer. Unfortunately, these known techniques are not readily adaptable to database usage metering, and moreover, are neither secure enough nor provide the security against database piracy that most database owners demand.

The prevention of unauthorized database usage becomes a huge problem whenever a stored database leaves the possession and control of the database owner. Computer program manufacturers lose millions of dollars each year to "pirates" who make unauthorized copies of software and distribute those copies for profit. Complex databases are often even more expensive to produce than programs, so that potential contributors of data base properties, as well as database owners themselves, may be extremely hesitant to permit electronic copies of their properties or databases to leave their control copies will be made. The copyright laws and contractual licensing agreements may deter, but will not prevent, unauthorized use and copying of database.

SUMMARY OF THE INVENTION

The present invention provides a database access system and method at a user site which permits authorized users to access and use the database and absolutely prevents unauthorized database use and copying. The present invention also provides a facility for measuring usage of the on-site database for the purpose of billing the user according to the amount he has used the database, and for periodically conveying the measured usage information to the database owner (or his agent) — while preventing the user from tampering with the measured usage information.

The invention solves fundamental media based electronic publishing issues including:

- Security of the information base. The present invention provides a code/decode Interlock System which includes both software and a tamper proof hardware module that prevents unauthorized and/or unmetered use of a protected information base. The present invention also supports a multi-level coded security access system limiting access to

various portions of a data base only to those individuals possessing the proper security code(s); and Ascertaining the degree of usage of the information base. The present invention stores, in one of several alternative forms of non-volatile memory, the dates and times that any files (or documents, sections, properties, etc.) are accessed and also records the amount of information read from each file into memory by the user.

With the present invention, a CD-ROM disk, for example, might contain all issues of 10 separate publications (technical, medical, business, etc.) going back for five years. Each publisher would be able to set the price for the use of its publication or publications and each publisher could then receive a "copyright royalty" return-on-investment based on the actual customer usage of the publishers' products. Therefore, publishers contributing more important, popular or costly to develop lexical information base properties could earn revenues commensurate with the market demands and pricing strategies for their products.

The present invention eliminates the necessity of determining how much of the net revenue of a CD information base product each contributing publisher should receive (currently an issue of considerable concern to publishers). The present invention also ensures the data security of information bases — a critical, frequently voiced, and previously unanswered problem causing considerable publisher anxiety. It would be quite difficult (requiring a high level of specialized expertise and costly high-powered computers) to "break" the hardware/software data security system provided by the present invention and copy material without being charged an appropriate fee.

Publishers can license their products at an exceptionally low initial cost to customers (i.e. for a \$25.00 initial fee instead of a \$1,000.00 or more annual fee). Low initial licensing fees would result from the usage auditing capability of the present invention and would allow new clients to experiment with the product at little or no risk. Similarly, customers who anticipate a low level usage of a given information base product may find the lower costs of a usage based fee schedule a practical and affordable justification to acquire a product that would otherwise not be purchased.

In sum, the present invention will:

1. Significantly accelerate market penetration of electronically published products due to substantially lower initial license costs;
2. Greatly enhance the ultimate market penetration of CD published products by making CD publications affordable to a much larger body of customers; and
3. Produce higher ultimate revenues per published disk from those customers who would otherwise have purchased a costlier version of the database product.

The security protection provided by the present invention will give publishers significant advantages in securing exclusive contracts for important publishing information base properties, since the invention provides the information base property contributors with:

1. Vastly superior copy protection security;
2. Ultimately greater revenue;
3. Publisher specific control over pricing; and
4. A return-on-investment commensurate with the market demand for their information base property.

In accordance with one important feature of the present invention, a storage medium stores the database in encrypted form, and also stores index information

which correlates portions of the encrypted database with index keys. The index information may itself be encrypted if desired. A host digital signal processor operatively connected to the storage medium is preprogrammed so as to generate a database access request, read the index information from the storage medium, identify (in accordance with the index information) the portions of the encrypted database which satisfy the access request, and read the identified encrypted database portions from the storage medium.

A secure decoder control logic device coupled to the host processor receives the encrypted database portions read by the host processor, decrypts portions of the encrypted database read by the host processor to produce corresponding decrypted information, and transmits the decrypted information back to the host processor. The decoder control logic device also measures the quantity of usage of and/or other parameters pertaining to the information decrypted by the decrypting device, and stores these measurements in a non-volatile (and in many cases tamperproof) memory device. The invention thus provides a detailed record of database usage — including a breakdown of usage of each file or "property" stored on a local storage medium. Additional decryption of database information can be prevented or disabled if more than a certain percentage of a database (or more than a specified contiguous portion of a database) has been copied by the user as an additional safeguard preventing unauthorized copying.

The system may further include means for preventing tampering with the memory device and/or the decoder control logic means.

In accordance with another important feature of the present invention, database usage information is stored at a user's site and is periodically communicated to a central billing facility. For example, the non-volatile memory device storing data indicating database usage may be housed in a replaceable module. Periodically, the user disconnects the module from his computer system and sends it to a centralized billing facility. At the centralized billing facility, the contents of the memory device are read and used to bill the user according to his database usage.

In accordance with yet another important aspect of the present invention, communications is periodically established between the user's site and a central facility for the purpose of telecommunicating database usage information stored at the user's site to the central facility.

In accordance with yet another important feature of the invention, the user is automatically prevented from decrypting the encrypted database after a predetermined event occurs (e.g., "expiration" of a memory module, or excessive database usage indicating copying attempts) unless the user has implemented an "antidote" (e.g., input secret information into his computer system and/or install a replacement component).

Because the database is stored in encrypted form (and/or the database directory is encrypted or otherwise coded), the only way to obtain useful database information is to decrypt portions of it using the tamperproof decrypting means of the invention. Safeguards may thus be used to prevent unauthorized database decryption.

Thus, the present invention resolves several fundamental problems that would otherwise impede the rate of growth of the CD-ROM and CDI electronic publishing markets. For example, it is a costly process to create

the core properties that may be incorporated into an information data base, and the structuring of the data base itself may, in some circumstances, be a costly effort. One way for data base preparers to recover the costs of constructing and maintaining a database is to charge a flat subscription or access fee to each user subscribing to use the database. If this is the only billing method used, however, infrequent users of the database may be discouraged from subscribing — because they would be asked to pay the same cost a frequent user pays. Furthermore, potential users may be hesitant to pay a significant one time or initial fee to acquire a technology or product with which they are unfamiliar.

With the present invention, a user will be able to pay (if so structured by the data base provider) according to his usage of the product and both the perceived risk, as well as — in lower usage environments — the high cost of the use of the technology, can be reduced or eliminated. Furthermore, since the present invention should accelerate the installed base and revenue growth rate for a given product, it may enable costs for even the high volume users to drop as well.

Moreover, database use can be measured simply by measuring the quantity of information which is decrypted. Other parameters relating to database usage (e.g., which databases and/or database subdivisions have been used; and the time, date and duration of use of each database and/or subdivision) may also be monitored and stored. The stored usage information can be periodically communicated to a centralized facility for billing the user in accordance with his database usage. Moreover, the user's on-site database access system can be designed to cease functioning unless the user installs a new component and/or inputs "secret" information — and the centralized facility can provide the user with such replacement components and/or secret information only when the user has paid his bill.

Because the invention provides a detailed record of which literary properties have been used and how much each property has been used, use payments paid by the user may be fairly apportioned to the property owners according to actual use of their respective properties. For example, if a user licenses a storage medium storing a library containing hundreds of different literary properties and then uses only two properties in the library, the owners of those two properties can be paid substantially all of the licensing fees charged to the user.

A free market system is thus maintained in an environment not otherwise susceptible to free market competition. Publishers and authors can be assured that they will receive incomes based on customer demand for their properties, and publishers can retain absolute control over pricing — despite the fact that the properties are being distributed on a storage medium along with hundreds of other properties. "Best sellers" can still be distinguished from unpopular works, and authors can be paid royalties based on consumer demand for their works.

This invention thus solves the fundamental CD and Optical publishing problem of how to provide end-users with disk libraries containing many different publications from different vendors. Different properties from different publishers have differing significances in the today's marketplace. These products have prices which each reflect vendor investment, product specific market demand, and other vendor product marketing considerations. The present invention allows each vendor to set a price for their product(s) carried on CD or other

media publications. The invention has an interlock system that prevents access to the non-volatile storage media (such as a CD-ROM disk) unless the user has contracted for the use of the disk and has a hardware plug-in module incorporating software.

When a customer makes use of stored data, the invention monitors which files are accessed and how much information is requested by the user to be displayed. In one embodiment of the present invention, information that is being reviewed or browsed may be distinguished from information that is read into a host computer for the purpose of copying, modifying, or telecommunicating, with different cost rates being applied to the different activities (so that, for example, the cost of browsing can be much less than the cost of copying or printing). Depending on the specific application and the nature of the user contract, the user might be required to:

1. Telephone the publisher once every three months, establishing a modem link over which a request is transmitted to telecommunicate back to the publisher the meter usage data; or

2. Mail to the publisher once every three months a removable EPROM module that contains the metered usage data.

The present invention thus prevents copying or browsing of a protected information base without adequate compensation to the publisher and its information base property (data) suppliers. Each supplier of information to an information base product receives a return on investment that reflects both the market demand for his specific property and the pricing and other marketing strategies that the supplier deems appropriate for his product.

The present invention allows very large numbers of customers to acquire library disks at very low initial costs, since the customer's billing can be largely based on usage, not simply possession of the library disk. As a result, potential customers, regardless of size or financing, will be able to maintain very broad based libraries on-site. If a given group regularly uses only a fraction of the information base, the group's users can still search the entire data base whenever appropriate. This means that most user billing is concentrated on those reference resources that the users frequently use, but an entire, comprehensive reference library extending beyond the user's frequent requirements is immediately available for use. A publisher will be in a much better position to provide large scale reference information base libraries. In many applications, the breadth and comprehensiveness of these encyclopedic libraries will encourage much more frequency use and a much larger body of users.

The present invention thus answers both the needs of a potentially very large customer base for low cost initial access to comprehensive digital disk base reference libraries, while at the same time maintaining supplies publisher control over pricing and guaranteeing an appropriate return on investment based on the customer's demand for their products.

The invention may be particularly attractive to the owners of the leading properties in a given vertical publishing market, since these owners are likely to be particularly sensitive to the issues of unauthorized access to and copying of their product, pricing of their product, and equitable return on the value of the contribution of their product to an information base library. These publishers are likely to greatly increase their revenues through participation in library publication

and distribution in accordance with the present invention — and the presence of such publishers in the marketplace will make it economically necessary (and feasible) for other publishers who have second tier properties to contribute to the same information base product.

The present invention may also include an optional security system which allows an organization to prevent usage of all or a portion of an information base unless the user enters his security code. Multiple levels of security codes can be supported to allow restriction of an individual's access according to his security authorization level.

There is significant value in using the present invention with certain types of non full-text information bases. For example, an electronic, CD disk containing comprehensive telephone white pages, telephone yellow pages, and as additional options, individual specific additional information (including estimated income level, publications received, job type and position, social security number, and other information that is compatible and legally available from one or more of the various mailing list companies) might be used with the present invention.

As a result of the present invention, the telephone operating companies providing directory listings can be compensated on the usage of their data base, while the mail order companies can also receive a revenue stream based on both usefulness of their data bases usefulness to customers and the extent of customer usage of their information. The present invention provides, for the first time, a context in which firms such as telephone operating companies and other information property suppliers can safely and profitably supply information for desk-top electronic information base products.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages of the present invention will be better and more completely understood by referring to the following detailed description of preferred embodiments in conjunction with the appended sheets of drawings, of which:

FIG. 1 is a schematic block diagram of a presently preferred exemplary embodiment of a database usage metering and protection system in accordance with the present invention;

FIG. 2 is a schematic block diagram of the information stored in the storage medium block shown in FIG. 1;

FIG. 3 is a more detailed schematic block diagram of the decoder/biller block shown in FIG. 1;

FIGS. 4a-4b are together a flow chart of the steps performed by the system shown in FIG. 1; and

FIG. 5 is a schematic block diagram of a further presently preferred exemplary embodiment of a database usage metering and protection system in accordance with the present invention, and

FIG. 6 is a flowchart of an overall method for receiving a return on investment from data bases at user sites.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

FIG. 1 is a schematic block diagram of a presently preferred exemplary embodiment of a database usage metering and protection system 10 in accordance with the present invention. System 10 includes three main blocks: a storage medium block 100, a host computer 200, and a decoder/biller block 300.

Predefined database(s) is (are) stored on storage medium 100 in encrypted form, and selective portions of the database(s) are read from the storage medium by host computer 200 (several different databases can be stored on the same medium, although the present invention in its simplest form uses only a single stored database which may contain multiple files, segments, "properties" or the like). Host computer 200 may be a computer dedicated to the task of accessing the stored databases, but need not be (for example, the host computer can be a general-purpose digital computer used to do a variety of different tasks).

Decoder/biller block 300 is connected to host computer 200, and performs at least two important functions. Decoder/biller 300 decrypts portions of the stored databases on a user need basis (e.g., after confirming the user has proper authority to access the databases) (see FIG. 6, block 904). Decoder/biller 300 also meters database usage, and generates usage information in a form which can periodically be conveyed to the owner of the databases (or his agent, e.g., a service company) (see FIG. 6, blocks 906-908). The usage information is typically used to calculate a database access fee the user is to be charged (see FIG. 6, blocks 910-914).

Decoder/biller block 300 may take the form of a hardware unit (or card) electrically connected to and located in proximity to (or within) host computer 200, or computer software executing on the host computer. Alternatively, decoder/billing block 300 might be located remotely to host computer 200 and communicate with the host computer via a data communications network or a telephone line.

Storage medium 100 is preferably some form of inexpensive mass digital information store (e.g., an optical disk, a bubble memory or a large hard disk or other fast transfer rate magnetic storage technology) prepared by the database owner and licensed to the user for use. CD-ROM, CDI, WORM, and other related optical/digital very large capacity storage modalities are now coming to the personal computer market and can be used for this purpose. These products are highly reliable, and very economically store hundred of megabytes up to many gigabytes of data.

For example, a CD-ROM diskette stores 550 megabytes of information on a single 12 centimeter laser diskette. CD-ROM technology now being released to the market will economically support up to eight parallel drives (4 gigabytes or 1,800,000 printed pages) and will access any desired sector in one second. In the next several years, technological advances should reduce access time to a second, and storage capacity will be doubled (430,000 pages per diskette and 3,600,000 pages on-line) if CD-ROM manufacturers decide to market double-sided disks and drives CD-ROM, CDI, and WORM products will be increasingly affordable over the next 30 months, with CD-ROM prices estimated to drop from \$800.00 to \$400.00 or less per drive, including controller, and OEM and volume prices estimated to drop to as low as \$175.00 per unit by 1990. With CD-ROM, WORM, and other optical/digital technologies, users can both purchase large scale information bases and also themselves easily build organization-specific information bases.

The database is preferably "preprocessed" and then stored onto medium 100. The type of preprocessing performed depends upon the database and the application, but typically includes creating an encrypted rendition

tion of the database and loading the encrypted rendition onto medium 100. One or more of the many sophisticated conventional data encryption schemes which presently exist can be used for encrypting the database. Preprocessing preferably also includes generating an index to the database and storing the index together with the encrypted version of the database on the storage medium 100. The index may or may not be encrypted.

The preprocessed database may be loaded onto storage medium 100 in a conventional fashion. For example, a "master" medium may be prepared, and then simply duplicated to yield a number of duplicate storage media 100. Storage of the entire preprocessed database (or databases) may require several storage medium units (i.e., several optical disks), each unit storing a part of the database. The database can index one or more databases each containing one or more files, documents or "properties" (the term "properties" referring to a literary or other textual work protected by copyright).

FIG. 2 shows one exemplary scheme for storing database information on medium 100. The information stored on medium 100 includes an index portion 102 and an encrypted database portion 104. Database portion 104 includes a plurality of predefined quantities, or "blocks", 106 of digital data. Each block 106 includes three information "fields" an index key field 108a; an encrypted database information field 108b; and a decryption key/error-checking field 108c.

Index portion 102, which may be encrypted, provides information used to translate a database access request into the addresses of one or more blocks 106. The contents of index portion 102 depends on the type of database stored on medium 100 and the type of operations which are to be performed on the database. For example, if word or string searching is to be provided, index portion 102 may include a list of all of the words contained in the database and the blocks 106 in which the listed words appear. Index portion 102 may alternately (or also) include a "table of contents" of the database and a designation of the blocks 106 covering each entry in the table. Other ways to index a database are known, and the present invention is not limited to any particular indexing scheme.

Index key 108a of each block 106 stores data which can be referenced in accordance with information stored in index information portion 102. Index key 108a may be explicit (e.g., a digital data word representing an indexing code or address) or implicit (e.g., physical "addresses" of storage medium 100 may themselves be used as indexing keys).

Encrypted database information fields 108b contains predetermined portions of the encrypted database. The size of these portions may be determined by the particular hardware and/or encryption techniques used, and is preferably (but need not be) fixed. If the nature of the database permits, logically-related information should be stored in the same blocks 106 (i.e., the database should be presorted and hierarchically organized) to reduce the number of accesses of storage medium 100 required to respond to a single user request. Techniques for organizing databases are known to those skilled in the art of information retrieval and database design and management.

Decryption key/error-checking field 108c performs two functions in the preferred embodiment. First, it provides conventional error checking (e.g. CRC or parity) information useful for detecting information

reading errors. Secondly, the field may provide information needed by sophisticated data decryption schemes to decrypt the information stored in associated field 108b. In many data decryption schemes, a decryption key word (which may itself be encrypted) carried with the encrypted data is used in conjunction with an additional data decryption key generated by the data decrypting device to decrypt the data. Field 108c may or may not be required depending upon the error checking and decryption schemes employed.

Host computer 200 contains resident software and hardware which provides an interface for all database transactions. Computer 200 includes one or more appropriate I/O handlers and associated hardware device drivers which permit the computer to read information from storage medium 100. Host computer 200 also includes appropriate data communications software and associated hardware which permits it to exchange data with decoder/biller block 300. The data communications pathway between host computer 200 and decoder/biller block 300 may be a shared data bus, a dedicated I/O channel, a shared data communications network, or the like.

When a user requests information from the database stored on storage medium 100, the computer program resident on computer 200 controls hardware of the computer to read the index information 102 stored on medium 100 in order to ascertain which database blocks 106 contain information specified by the user request. The computer program then controls host computer 200 to load one or more blocks 106 of the stored database information into the host computer memory. The host computer 200 then, under software control, strips off the contents of encrypted fields 108b from the blocks of information now resident in its memory (along with some or all of the contents of decryption key/CRC field 108c) and sends some or all of this information to the decoder/biller block 300 for processing.

Because the index portion 102 is not encrypted, host computer 200 can manipulate the index information without involving decoder/biller block 300. Although this is an important advantage in some applications (since the user is permitted to "browse" through the index "for free"), other applications may demand a level of security which is compromised by providing an unencrypted index. For example, unencrypted, very complete indexes might be used to reconstruct significant portions of the database itself. It may therefore be desirable to encrypt index portion 102 as well as database portion 104 to provide higher security.

If index portion 102 is encrypted, it must be decrypted before a user can make selections from it or otherwise use it to locate blocks 106. Decryption of index portion 102 should be performed in a secure environment (such as in decoder/biller block 300, or in a dedicated "browsing workstation" to be discussed in connection with FIG. 5). Alternatively, decoder/biller block 300 may temporarily provide host computer 200 with the decryption key information needed to decrypt index portion 102 (the index portion may be encrypted using an encryption technique which is different from the one used to encrypt database portion 104), and the host computer can decrypt sections of the index portion as needed by the user.

In one possible permutation of the invention, neither the database nor the index stored on medium 100 is "encrypted" using a formal encryption algorithm, but instead, the manner in which the database and/or the

index is stored on the storage medium is itself used to make information incoherent unless it is read from the medium using a predefined access algorithm.

For example, records of the database may be non-contiguously stored on medium in a pseudo-random order, so that sequential reading of records produces only incoherent information. An index stored on medium 100 contains the information needed to locate logically sequential database records. This index ("directory map") may also be in some way "scrambled" (for example, encrypted using formal encryption techniques, perhaps simply incomplete so that it must be supplemented with information and/or algorithms contained in decoder/biller block 300, or another scheme can be used to properly interpret the directory map, directory map interpretation being necessary to determine the locations on medium 100 of the components of a given database or other "property"). Different index scrambling schemes can be used for different copies of storage media 100 to prevent development of a "universal" de-scrambling device or algorithm.

Decoder/biller block 300 measures the amount and/or type of information sent to it for decryption and stores information indicating database usage over time from such measured amounts. Decoder/biller block 300 stores all necessary billing and usage information in a protected, non-volatile memory device (or in a protected, non-volatile storage facility within the host computer 200) for later retrieval and use in calculating database usage fees.

Because the database information read from medium 100 is useless unless it is first decrypted, and decoder/biller block 300 is the only portion of system 10 which is capable of decrypting the encrypted database information, the decoder/biller block can accurately meter the amount and nature of data accessed from the stored database (e.g., by counting the number of blocks 106 which are encrypted, determining the group of logically related information ("property") stored on medium 100 which is logically associated with the data being decrypted, and/or determining other convenient parameters indicating the quantity and/or identity of data which is decrypted). Decoder/biller block 300 decrypts the information sent to it, and returns the decrypted information to host computer 200 for display, storage, printing, telecommunications, or the like (or otherwise makes the decrypted information available to the user).

FIG. 3 is a more detailed schematic diagram of the decoder/biller block 300 shown in FIG. 1. Block 300 includes the following: a tamper-proof mechanism 302; a data connector 304 for connection to the host computer 200; a data connector 306 for connection to an off-site service decryption logic 310; interface logic 312; a non-volatile memory 314; decoder control logic 316; and a real-time clock/calendar 318.

Tamper-proof mechanism 302 prevents unauthorized persons from electronically or mechanically tampering with decoder/biller block 300, and preferably includes both mechanical and electronic safeguards. For example, the physical enclosure which encapsulates the components of block 300 should prevent unauthorized individuals from accessing the enclosed components. The components can be epoxied or potted if desired, and/or the enclosure can be provided with a mechanical seal which clearly evidences any tampering.

Another safeguard against tampering can be provided by implementing one of more of functional blocks

308-318 in the form of a custom integrated circuit. Such custom integrated circuits are not easily reproducible by an unauthorized person, nor could functional equivalents be designed ("black-boxed") so long as the techniques used to encrypt and decrypt the database are sophisticated. This level of data encryption sophistication is well within present technology.

Connector 304 and interface logic 308 communicate data between decoder/biller block 300 and host computer 200. Interface logic 308 includes conventional electronics which interface host computer 200 with decoder control logic 316. Interface logic 308 is electronically connected to physical electronic connector 304, which in turn is connected to a mating connector of host computer 200.

The exact configuration of interface logic 308 and connector 304 depends upon the nature of host computer 200 and sort of data communications pathway desired. For example, in one exemplary arrangement, connector 304 comprises a host computer bus connector (connected to the main bus of host computer 200 and addressed directly by the host computer processor) and interface logic 308 comprises a bus interface. Of course, connector 304 could comprise a standard RS-232 port connector and interface logic 308 could comprise conventional port interface logic — or the interface logic could comprise a communications controller (e.g., a data communications network controller or a modem) and the connector 304 could be a standard communications connector (if decoder/biller block 300 were located remotely from host computer 200).

Other communications connectors and/or ports might be used for connector 304, the specific arrangement used being chosen based on the application, convenient performance and/or cost. Other possible arrangements, including placing the decoder/biller block 300 into the same housing containing the drive which accesses medium 100, or connected to (or actually connected as part of) cabling connecting the drive for medium 100 to host computer 200, can be used.

Decoder control logic 316 preferably includes a conventional microprocessor pre-programmed with a predetermined control computer program, but might be implemented in other ways (e.g., as a discrete digital logic sequential state machine). Decoder control logic 316 controls all of the functions of decoder/biller block 300 in the preferred embodiment. Decoder control logic 316 also monitors database usage, produces digital data indicating the amount of such usage, and stores this data in non-volatile memory 314 for later retrieval (e.g., by a service company or the database owner).

Real time clock/calendar 318 permits database usage metering to indicate the time and date of each usage and the duration of usage, thus providing an important audit tool for both customers and the service company. In addition, this real-time clock/calendar 318 can be pre-programmed to allow the user to access certain databases only at pre-programmed times (e.g., by limiting access for given user security access codes).

Interface logic 312 and connector 306 may be used to communicate data with an off-site facility, such as the centralized computer of the database owner or a service company which handles periodic database usage billing. In one exemplary embodiment, connector 306 includes a standard telephone connector and interface logic 312 includes a standard modem. If desired, connectors 304 and 306 may comprise the same connector, and inter-

face logic 308 and interface logic 312 may comprise the same components.

Database decryption logic 310 takes input digital data signals provided to it by decoder control logic 316 (these signals representing encrypted digital data read by host computer 200 from storage medium 100 and passed to the decoder control logic via connector 304 and interface logic 308), decrypts these digital data signals using a predefined decryption algorithm, and outputs decrypted data signals to the decoder control logic for display, printing, and the like. One or several different predefined decryption algorithms can be stored in (or hardwired within) decryption logic 310, and additional decryption algorithms can be downloaded into the decoder/biller block 300 as needed or required via interface logic 312.

Many conventional methods of encrypting/decrypting data are known, spanning from simple lookup tables to complex mathematical algorithms. The method of data encryption/decryption used depends on the amount of extra computer processing overhead and data storage space that the application will allow. It is not uncommon for substantial overhead to be needed to handle encrypted data.

To install system 10, storage medium 100 (along with its associated drive/access device) is connected to host computer 200, and decoder/biller 300 is also connected to the host computer port and/or bus (by connecting connector 304 as described). A non-volatile memory 314 is provided which has been preloaded with the following information (or is loaded upon installation):

- (a) database key(s) and/or user password(s);
- (b) billing rates (optional — may be performed by the data base owner at his own facility);
- (c) expiration date and "antidote" information; and
- (d) user identification(s)/security levels (if desired).

FIGS. 4(A)–4(B) are together a high-level flowchart of the routine 400 performed by system 10 to access a portion of the stored database.

To access database information, the user causes host computer 200 to execute software resident within it which permits the user to formulate a database access request (block 402). As discussed above, the nature of the access request depends on the nature of the database and the needs of the user. Most users require the ability to perform lexical database searches (i.e., searches for words, strings, and the like). However, other methods of accessing information are also possible. For example, if the database is a literary novel, the user's access request might be a chapter number and/or page number. Personal Library Software, Inc. of Bethesda, Maryland, offers advanced indexing software technology which allows a user to perform both keyword and topical searches (contrasting with other commercial products, which are limited to keyword searching techniques). Personal Library software can be used to great advantage with the present invention.

The user then inputs an access request (block 404) using a keyboard or other standard I/O device connected to host computer 200. In response to the user's access request, host computer 200 accesses index portion 102 stored on medium 100 and obtains from the index portion the addresses of (or index keys corresponding to) each block 106 of the encrypted database which satisfies the user's access request (block 406) (index portion decryption is performed at this time if necessary). Host computer 200 then reads the appropriate block(s) 106 of the encrypted database from storage

medium 100 and stores these blocks of information into its own internal random access memory (block 408).

System 10 may require the user to input identification and/or password information along with his access request (block 404). System 10 checks the authority of the user to access the database by transmitting the input ID/password information to decoder/biller block 300 for comparison with a list of authorized IDs/passwords stored in memory 314 (block 410). If decoder/biller block decoder control logic 316 denies authorization to continue with database access (because the inputted user information is incorrect, because the access request cannot be performed at the current time/date, etc.) (block 412), the decoder/biller block refuses to decrypt any data sent to it (block 414) — and may cease communicating with the host computer 200, and/or simply ignore any encrypted information the host computer sends it. While encrypted database information is already present in the memory of host computer 200, this encrypted information is incoherent and cannot be used for any useful purpose.

On the other hand, if decoder control logic 316 of decoder/biller 300 grants authority to proceed (block 412), the decoder control logic begins a "billing cycle", and stores information logging the billing cycle into non-volatile memory 314 (block 416). The information stored in memory 314 may include: (a) the name of the database file being accessed; (b) the section of the database being accessed (name, "property designation", file name, or other identification information); (c) the identification of the user accessing the database; and (d) the date and time the database access begins.

The information stored in non-volatile memory 314 may thus be used to create an "audit trail" which tracks different users (or groups of users) and their database usages. Special use passwords may be required to access selected databases, and actual use of all databases may be verified later from the information stored in memory 314. Such stored information is extremely valuable not only to help detect unmonitored database use, but also to allow detailed bills to be generated and to help determine which users among multiple users are responsible for generating usage charges. Such a detailed audit trail can be used to allow publishers and users to determine the detailed activities of users. This information can be used by users to determine what they are being charged for. The audit trail information can also be used by publishers and property owners to conduct marketing surveys — providing more detailed information about user demographics and information use than is presently available.

In addition, it may be desirable to code storage medium 100 (or particular databases or files stored on the medium) with unique (e.g., randomly-generated) user passwords by embedding secret password information in the database information. Non-volatile memory 314 can store information which matches the code associated with the particular copy of the storage medium licensed to a particular user. This coded information can be encrypted, and coding schemes and/or coded information may be changed periodically. Different users can be assigned different codes to prevent users from exchanging or sharing storage media 100.

This additional security feature also impedes the use of unauthorized decoder units (e.g., clandestine units manufactured to be similar to block 300). Such unauthorized units would not be equipped with the correct coded information, and even if they were, would work

for only one similarly coded storage medium (or for only one or a few databases stored on a particular storage medium). The coding of storage medium 100 with embedded, user-identifying codes would also help to identify how any unauthorized copies of the database information came into being, since the coded information would be embedded in the database information itself and would thus also be present in any copies made from an original. Users found in this manner to be involved in copyright infringement could be penalized appropriately under the civil and criminal penalties of the copyright law, as well as for breach of their contractual obligations.

Decoder control logic 316 also is enabled at this time to begin (a) decrypting information sent to it by host computer 200 and (b) sending the decrypted information back to the host computer (block 418). Decoder control logic 316 meters the quantity and/or other usage parameters of data which is decrypted, and stores this usage information into non-volatile memory 314 along with the other billing information (block 420) (the decoder control logic may store quantity information directly into the memory, or may first convert it to billing information taking into account, for example, the cost of using the database file being accessed). This process continues until the user's request has been satisfied (as tested for by block 422).

The user can be billed an annual fee for unlimited use of some databases or database properties, and billed only for actual use of other databases or database properties. In this way, the user can pay a flat fee for the databases, or specific database properties or "books", he uses most often, and yet have access on a "pay-as-you-go" basis to other databases which he might use occasionally but not enough to justify paying the cost for unlimited use. This billing method provides the user with database resources he might not otherwise be able to afford, and also stimulates use of databases which are not used often but are nevertheless extremely valuable at times.

The specific steps performed to decrypt data (block 418) depends on the particular data encryption/decryption scheme used. Host computer 200 transmits encrypted data in predetermined quantities (e.g., fixed-length blocks) to interface logic 308 via connector 304 in the preferred embodiment. Interface logic 308 communicates this encrypted data to decoder control logic 316, which communicates it to data encryption/decryption logic 310. Logic 310 translates the encrypted data into intelligible information using a predetermined conventional decryption algorithm, and communicates the decrypted data back to decoder control logic 316. Decoder control logic 316 then communicates the decrypted data to host computer 200 via interface logic 308 and connector 304.

The database access program resident in the host computer then controls the host computer to display and/or print the decrypted information. If desired, the program resident in the host computer 200 can prevent the user from doing anything other than displaying (and/or printing) the decrypted data. Alternatively, this program may permit the user to manipulate the decrypted text (e.g., store the data in a disk file or in the memory of the host computer) to permit the user to browse through full-text data at his leisure and/or to use this data for word processing, telecommunicating, or the like.

Decoder control logic 316 meters database usage (block 420) by, for example, measuring the amount of information which is decrypted (e.g., by counting the number of fixed-length blocks which are decrypted; determining the source documents the decrypted information is associated with; and measuring the time, date and/or duration of access of the decrypted information). Control logic 316 may also record other billing information, such as the length of the database file being opened. Control logic 316 may be arranged to recognize the names or other designations of subsections of the database being accessed, allowing for different billing rates depending on the type or supplier of the information (so that use of more expensive databases can be billed at higher rates).

It may be desirable to not bill users for simply searching through the database (or at least, not bill at the full rate), but to bill only or at a higher rate for data that is decrypted and displayed, printed or communicated. It is for this reason that the database index is not itself encrypted in one embodiment — so that the user can browse through the index “for free” (or at a lower charge). As mentioned previously, however, it may be desirable in some instances to provide additional security by encrypting the index as well as the database. If decoder/biller block 300 decrypts the index, it can meter index usage and store this usage information into non-volatile memory 314 — thus permitting the user to be billed for index browsing at comparatively low rates. A dedicated “browsing terminal” (to be discussed shortly) may be used in some applications to provide a secure environment in which browsing can occur and billed at a rate which may differ from that for database information usage (e.g., printing, telecommunicating, copying, etc.).

After the user's access request has been satisfied (as tested for by block 422), the decoder control logic stores, into non-volatile memory 314, the time the user finishes accessing the database. (block 424). The resident program then allows the user to input another access request (using the same or different database) (block 426). If the user does input another access request, the steps of blocks 404-426 are performed again (with blocks 416, 420 and 424 causing an additional billing log entry to be stored in memory 314).

The information stored in memory 314 is periodically communicated to the service company and used to bill the user for database usage. In one exemplary embodiment, memory 314 is housed in a storage module 314a which is easily separable from system 10. Periodically, the user disconnects memory module 314 from decoder/biller block 300, mails the module to the service company, and installs an alternative replacement module (the “next” module) into system 10. Decoder control logic 316 disables data decryption unless a module 314a is connected to it (and perhaps also when the control logic has determined the non-volatile storage area is nearly full).

In another embodiment, communications between decoder/biller block 300 and the service company is periodically established for the purpose of downloading the contents of memory 314 to the service company billing computer. If connector 306 and programming interface logic 312 comprise a conventional standard telephone connector and associated modem, such communications can be established over standard telephone lines. The information stored in memory 314 is transmitted over the telephone line to the service company

computer, and the service company computer then transmits commands which control decoder control logic 316 to reset the memory. In addition the service company can establish communications with decoder/biller block 300 to monitor use of the databases stored on medium 100 (and detect misuse and unauthorized use). The service company may also control decoder/biller block 300 remotely (e.g., to disable it from operating if customer fails to pay his bill).

System 10 may include an enabling/disabling mechanism which prevents a user from accessing the stored database information if he fails to pay his bill. For example, in the embodiment discussed above having a separable memory module 314a, the service company can refuse to mail the user a replacement module until all outstanding balances are paid. If the customer fails to pay his bill, he will eventually fill up the memory module he has installed, causing decoder control logic 316 to disable data decryption (or alternatively, the modules 314a can be electronically date-coded, and the decoder control logic can refuse to permit decryption to be performed when the module date code is determined to be prior to the current date generated by real time clock/calendar 318).

Decoder control logic 316 can be disabled from operating if the real time clock ever ceases to operate (for example, the clock may be battery powered and the battery might go dead after a year or so if scheduled preventive maintenance is not performed). Once the real time clock is repaired, a communications link can be established between decoder/biller block 300 and the central facility. The central facility can then read the contents of non-volatile memory 314. If no suspicious or unauthorized activities have occurred, the central facility can reset real time clock 318 or check a locally set real time clock to permit normal database decoding operations to resume.

Another arrangement can control decoder control logic 316 to periodically, automatically change authorized passwords — and the service company can refuse to tell the customer the new passwords until the customer has paid his bill.

Alternatively or in addition to the arrangements discussed above, system 10 may be provided with an automatic “self-destruct” mechanism which automatically “destroys” a critical part of the system (e.g., the information stored on medium 100, or the password table stored in non-volatile memory 314) at a preset real time deadline (timed by real time clock/calendar 318) unless the customer implements an “antidote” (e.g., inputs a series of secret code words) prior to the deadline. The service company can provide antidote instructions only to customers who have paid their bills. This automatic “self-destruct” mechanism can also be activated whenever the customer exceeds a predetermined maximum (and/or minimum) usage limit (so as to prevent a customer from running up a huge bill, from attempting to decrypt and store substantial portions of the unencrypted database, or from continuing to use the database in the unlikely event that he has successfully prevented the logging of usage information). If additional protection against database piracy is desired, the automatic “self-destruct” mechanism can also be activated whenever the user attempts to access, in one session or over a number of different sessions or within a given time frame, more than a certain percentage of a given database and/or more than a certain number of contiguous blocks of (or logically related records or other

subdivisions of) the same database. A permanent record of the blocks (records or other subdivisions) which have been accessed may be retained in non-volatile memory 314 so that the user can be prevented from copying an excessive amount or selected database properties or segments over a period determined by the database owner.

It may also be desirable to enable the user to program parameters stored in non-volatile memory 314 which limits the user's own use of database information stored on medium 100. The routine shown in FIGS. 4(A)–4(B) can provide a user interface with decoder/biller block 300 which permits a user to optionally store, in a user-accessible file within memory 314, information representing ceilings on database usage or cost of usage over a period of time (e.g., a maximum monthly duration or cost for database usage, limitations on the type of information which can be decrypted, etc.). Decoder/biller block 300 keeps a running total of the parameter(s) the user has specified, and ceases decrypting database information if the total exceeds the user-specified parameter value. This feature permits the user to budget his database use, and is especially valuable in a business environment — since it permits an organization to directly limit the cost of database access by employees to an amount selected by the organization.

Although the embodiment shown in FIG. 1 is particularly suited for installation at a customer site, some applications might necessitate that decoder/biller block 300 and storage medium 100 be operated remotely to the customer site and communicate information to the customer via a communications link (e.g., a standard telephone line). In this "direct connect decryption" mode of operation, data decryption is performed at a central facility of the service company. Since only a small portion of the database is decrypted at any one time, a telephone line provides sufficient bandwidth to transmit the decrypted data at rates suitable for display by the customer's computer.

Using the "direct connect" mode, there is no need for periodic exchange of service storage modules or for pre-scheduled periodic communications with the local host computer. Billing data could be accrued in real time, and the service company could disconnect or change the service of a customer at any time. Database updating is also simplified, and current information or changing data is always at hand (since it can be automatically included in a user database search). Moreover, the user can use just about any kind of computer to access the service company central facility. Furthermore, the connect time charges for communication networks are becoming more competitive in price, making this "direct connect" mode attractive for some applications.

The chief disadvantages of this "direct connect" approach are: Database access speed is much slower than in the locally-installed embodiment discussed above (because of the shared nature of the central facility and because of the relatively low data transmission rate of standard telephone lines); communications costs are much greater; and the service company must purchase and operate an expensive multi-user computer facility.

The "direct connect" and the locally stored database features might be used together in some applications. For example, the bulk of a database can be stored on and accessed locally from a local storage medium 100. Database update file information can be stored and updated at a remote centralized facility and accessed via

a telecommunications link to provide extremely current information in addition to the "older" information provided on-site.

There are thus both advantages and disadvantages to the "direct connect" mode. This mode may be offered as an option for users who require up-to-the-minute updated databases.

Once data is decrypted and stored into the memory of host computer 200 (e.g., for searching or manipulation rather than simply for display), it is susceptible to being intercepted by a "pirate" intercept program. System 10 bills for the data which is decrypted (so that the user would run up a huge bill if he tried to copy a large portion of a database). Nevertheless, it may be desirable in some applications to restrict the manner in which a customer can use decrypted data, while at the same time not restricting manipulations (e.g., browsing) of the decrypted data.

For example, keyword searching does not require a data image of the database (rather, it is most efficiently performed using index information 102). However, other search techniques (e.g., final "zooming in" of the information being searched for) may require manipulation of a data image. It may be desirable to absolutely prevent the user from copying the decrypted data image information. However, the user should be able to manipulate data images in other ways (e.g., by browsing through full-text data and the like). It may be impossible to impose such restrictions on data stored in the user's own host computer 200 (or the user may be able to easily defeat such restrictions once imposed through skillful programming techniques).

FIG. 5 is a block diagram of an alternate embodiment of a database usage metering and protection system 500 in accordance with the present invention. The FIG. 5 embodiment includes a dedicated independent hardware unit ("browsing workstation") 501, which can either act as a "stand-alone" or be designed to interface with additional data processing components.

Browsing workstation 501 in the preferred embodiment includes a proprietary, single-board computer 502 connected to a dedicated proprietary display station 504 having a secure environment. Computer 502 includes a bus connector 506, a host interface 508, a CPU 510, a volatile, protected memory 512, a non-volatile memory 513, and a display driver 514. Computer 502 is enclosed in a tamper-proof enclosure 516 to completely prevent access to its internal components except by authorized service personnel.

Computer 502 performs the decryption and billing functions discussed previously, and then stores the decrypted data into its own memory 512. This arrangement allows the user to review ("browse") the information (on dedicated display station 504) prior to sending desired information to his host computer (via interface 508 and connector 506) for printing or other use. Thus, the decrypted database data image is first stored and manipulated by computer 502. The user can be billed at one rate for browsing through or otherwise manipulating data in computer 502, and billed at a higher rate for transferring data to his host computer (from which the data can be printed, stored, outputted, or telecommunicated to other computers and users).

The user can evaluate the data while it is resident in computer memory 512 (via display station 504) in order to decide whether or not he really wants the information transferred to his own host computer. In this way, very different billing rates can be provided for (a)

browsing large amounts of full-text information and (b) actual use of information in the host computer (e.g., for word processing, telecommunications, printing, etc.).

Browsing workstation 501 may share some of the hardware and/or software of a host computer in order to reduce hardware costs — so long as information security is not significantly compromised. For example, one of the workstations normally connected to the host computer and its associated driver might be used in lieu of dedicated display station 504 and display driver 514 if there is little or no possibility that the user could copy a significant part of a database by reading information produced by the host computer display driver while browsing is in progress.

In a further embodiment, sophisticated software (not susceptible to manipulation or other misuse) could be temporarily loaded into the host computer (e.g., from storage medium 100) and executed to provide the functionality of some or all of the hardware "blocks" shown in FIGS. 3 or 5. Such software might use the security system provided by the host computer (and/or sophisticated techniques which are difficult to discover and "break") to create a protected environment within the host computer itself for decryption of database information and non-volatile storage of database usage information which may be adequately secure for various applications.

For example, although it may be undesirable to permit data type decryption key information to reside in the host computer permanently, the decryption key information can be temporarily provided by a protected memory device to the host computer. The host computer may then decrypt database information using the decryption key information, and destroy the key information after use. The host computer may decrypt database information "on the fly" and not retain much encrypted or decrypted information in memory at any one time to help prevent copying.

Although a dedicated hardware/software system typically provides the best assurance against tampering, techniques which may be implemented in software executing on a non-dedicated system may provide sufficient tamper resistance for some applications. For example, secure program control and usage information can be stored on a floppy disk which is accessed via the disk drive of a general-purpose non-dedicated personal computer. A non-volatile memory and logic device connected to the personal computer may (in conjunction with the secure program control software executing on the computer and/or a hardware controller connected to the computer) control and monitor the position of the read/write head of the disk drive, store the current head position in the non-volatile memory, and supervise execution of the secure program control software. Database usage information may be gathered by the program control software and stored on the floppy disk. Any attempts to tamper with the floppy disk which alters the last read/write head position may cause a warning message to be stored on the floppy disk in a database audit trail section of the disk (possibly along with cumulative messages indicating previous such occurrences) and may also result in destruction and/or disablement of the secure program control software.

While the present invention has been described with what is presently considered to be the most practical and preferred embodiments, it is to be understood that the appended claims are not to be limited to the dis-

closed embodiments, but on the contrary, are intended to cover modifications, variations, and/or equivalent arrangements which retain any of the novel features and advantages of this invention.

What is claimed is:

1. A secure data base access system comprising: at least one storage medium storing at least one encrypted textual database component and at least one index associated with said component; input means for providing database index search criteria in response to user input; searching means operatively connected to said at least one storage medium and to said input means for searching said database, including means for referencing said index based on said search criteria, for identifying portions of said encrypted database in response to said index referencing, and for reading one of (a) all of said identified database portions, and (b) desired ones of said identified database portions from said at least one storage medium;
- means, connected to said searching means, for decrypting at least one portion of said read encrypted information; and
- control means operatively connected to said decrypting means for metering usage of information decrypted by said decrypting means and for telecommunications said signals representing said usage to a remote location over a telecommunications network.
2. A system as in claim 1 wherein said control means measures the number of contiguous blocks of said textual information decrypted by said decrypting means and prevents said decrypting means from decrypting more than a certain number of said contiguous blocks.
3. A system as in claim 1 wherein said control means measures the time duration over which at least one (a) of said searching means processes said information, and (b) said information is decrypted, and wherein said metering means includes means for storing said measured time duration.
4. A system as in claim 1 or 2 wherein said metering means measures the duration of usage of said decrypted information.
5. A system as in claim 1 or 2 wherein said metering means includes means for storing said usage information.
6. A system as in claim 1 wherein: said stored encrypted information has logically-related segments; and said control means determines which logically-related segments said selected portions are associated with.
7. A system as in claim 1 wherein: said medium stores plural textual databases; and said control means meters usage of less than all of said databases.
8. A system as in claim 1 wherein said control means comprises: a physically separable non-volatile memory device; and monitoring means, connected to said decrypting means and also disengageably connected to said memory device, for measuring the quantity of information decrypted by said decrypting means and for storing indicia of said measured quantity in said memory device.

9. A system as in claim 8 wherein said monitoring means disables said decrypting means from operating unless said memory device is operatively engaged thereto.

10. A system as in claim 1 wherein said control means comprises:

memory means for storing usage information; monitoring means, operatively connected to said decrypting means and also connected to said memory means, for metering at least one of (a) the quantity of information decrypted by said decrypting means, (b) an identification of a subset of information stored on said medium containing said identified portions, and (c) duration of at least one of searching, identifying, decrypting, reading and using of said database portions, for generating signals indicating the result of said metering, and storing said generated signals in said memory means; and

means operatively connected to said decrypting means and to said memory means for preventing said decrypting means from decrypting information whenever said metered indicating signals are not successfully stored in said memory means.

11. A system as in claim 1 wherein said control and communicating means includes:

a memory; and monitoring means, operatively connected to said decrypting means and to a communications network, for monitoring the quantity of at least one of: (a) information decrypted by said decrypting means and (b) information identified by said searching means, for controlling said signal communicating means to communicate an indication of said monitoring to a billing facility over said communications network.

12. A system as in claim 11 wherein said monitoring means also determines identifying characteristics of at least one of (a) said decrypted portions and (b) said identified portions and controls said signal communicating means to communicate said identifying characteristics to said billing facility.

13. A system as in claim 1 wherein:

said at least one storage medium also stores unencrypted index information correlating unencrypted search information with portions of said encrypted database.

14. A system as in claim 1 wherein:

said at least one storage medium also stores encrypted index information correlating search information with portions of said encrypted database.

15. A system as in claim 1 further including:

a first memory means, operatively connected to said decrypting means, for storing said decrypted information; and

a second memory means, operatively connected to said metering and communicating means and different from said first memory means, for storing said metered usage.

16. A secure database access system as in claim 1 wherein said control means includes means for using said decrypted information and means for metering the duration over which at least one of: (a) said decrypting means decrypts said read encrypted information, (b) said using means uses said decrypted information and (c) said searching means searches said at least one database.

17. A system as in claim 1 wherein:

said at least one database stored by said at least one storage medium is divided into plural discrete subdivisions;

said control means includes means for determining the subdivisions said selected portions are derived from; and

said metering means includes means for telecommunicating signals indicating said determined subdivisions.

18. A system as in claim 1 wherein said control means measures the duration of usage of said decrypted information, and wherein said metering means includes means for storing said measured duration.

19. A system as in claim 1 wherein said control means stores said identifications and/or measurements, said control means including means for inhibiting said decrypting means from further decrypting said database whenever said memory device becomes filled and means for resetting said memory device in response to said certain information received from said distant location.

20. A secure data base access system comprising:

at least one storage medium storing at least one textual database component and at least one index associated with said component;

input means for providing database index search criteria in response to user input;

searching means connected to said at least one storage medium and to said input means for searching said database, including reading means for referencing said index based on said search criteria, for identifying portions of said database in response to said index referencing, and for reading one of (a) all of said identified database portions, and (b) desired ones of said identified database portions from said at least one storage medium; and

control means connected to said reading means for metering usage of information read by said reading means and for preventing said reading means from reading more than at least one predetermined percentage of said at least one database in response to said meter usage,

wherein:

said at least one storage medium stores at least one scrambled directory of the location of the contents of at least one of (a) said at least one index, and (b) said at least one database as stored on said at least one medium; and

said reading means includes means for descrambling said at least one scrambled directory and reading said identified database portions from said medium in a manner determined by said descrambled at least one directory.

21. A secure data base access system comprising:

at least one storage medium storing at least one textual database in encrypted form, said at least one database including at least one collection of textual information, said at least one storage medium also storing index information, said index information correlating portions of said at least one encrypted database with search information;

at least one host signal processor, operatively connected to said at least one storage medium, said at least one processor preprogrammed so as to: (a) accept search criteria in response to user input thereto, (b) search said index information, (c) identify, in accordance with said search of index information, the portions of said at least one encrypted

database which satisfy said search criteria, and (d) read at least one of said identified encrypted database portions from said at least one storage medium;

means for decrypting at least one read portion of said encrypted at least one database to produce corresponding decrypted information; and decoder control logic means, coupled to said host at least one processor, and said decrypting means, for measuring the percentage of at least one information collection decrypted by said decrypting means, said decoder control logic means including means for preventing decryption of more than at least one predetermined percentage of said at least one information collection.

22. A method of providing information comprising the steps of:

- (1) providing at least one storage medium storing encrypted textual database information thereon;
- (2) searching said encrypted information to identify at least one portion of said encrypted information;
- (3) reading at least one of said identified portions from said at least one storage medium;
- (4) decrypting at least one of said read portions;
- (5) metering the usage of portions decrypted by decrypting step; and
- (6) calculating a usage fee in response to said measured usage.

23. A method as in claim 22 wherein said metering step (5) includes the step of measuring the quantity of information decrypted by said decrypting step.

24. A method as in claim 22 wherein:

said at least one storage medium stores said encrypted information in blocks of predetermined length; and said metering step includes the step of counting the number of said blocks of information decrypted by said decrypting step.

25. A method as in claim 22 wherein said metering step includes the step of determining the time at which said decrypting step decrypts said information.

26. A method as in claim 22 wherein:

said at least one storage medium stores plural discrete collections of encrypted information; said method further includes the step of selecting at least one of said plurality collections; and said metering step includes the step of storing signals indicating the usage of said selected at least one collection.

27. A method of accessing a database comprising the steps of:

storing encrypted text database information in digital form on a random access nonvolatile storage device;

searching for and retrieving portions of said stored database based on search criteria at least in part determined by user input;

determining the quantity of one of (a) all of said retrieved database portions and (b) only desired ones of said retrieved database portions ready said searching and retrieving step;

metering information representing said determined quantity and storing said metered information in a further non-volatile storage device different from said first-mentioned storage device; and periodically conveying said stored quantity-representing information to a location remote thereto.

28. A method of securing access to a database comprising the steps of:

providing at least one random access storage medium having at least one database in encrypted form stored thereon and also having index information stored thereon, said index information correlating portions of said encrypted at least one database with unencrypted search information;

generating search information;

searching said index information for specific portions of said encrypted at least one database which satisfy said generated search information;

decrypting at least one of said specific portions of said encrypted at least one database to produce corresponding decrypted information;

measuring the quantity of information decrypted by said decrypting step; and

inhibiting said decrypting step from decrypting more than a certain percentage of said at least one database in response to said measured quantity.

29. A method of securing access to a database comprising the steps of:

providing a random access mass storage means having at least one database in encrypted form stored thereon and also having index information correlating portions of said at least one encrypted database with index information stored thereon;

providing a database search request determined at least in part by user input;

searching said index information from said storage means with a digital signal processing means of the type capable of displaying data and also capable of performing at least one of the additional functions of copying, storing, printing and communicating data and searching, in accordance with said index information, for specific portions of said at least one encrypted database which corresponds to said search information;

decrypting at least one of said specific portions of said at least one encrypted database to produce corresponding decrypted information;

selectively enabling further processing of said decrypted information by at least one of said additional functions;

determining a cost for performing said displaying step in response to at least one first cost rate and at least one of (a) the quantity of information displayed and (b) the time duration over which at least one of said above-mentioned steps is performed;

determining a further cost in response to at least one second cost rate different from said first rate and in response to one of: (a) the quantity of information which is further processed by said additional functions and (b) the time duration of which at least one of the said additional functions is performed; and storing at least one of said determined costs.

30. A method of securing access to a database comprising the steps of:

providing a random access mass storage means having at least one database in encrypted form stored thereon and also having index information correlating portions of said at least one encrypted database with index information stored thereon;

providing a database search request determined at least in part by user input;

searching said index information on said storage means with a digital signal processing means of the type capable of displaying data and also capable of performing at least one of the additional functions of copying, storing, printing and communicating

data and searching, in accordance with said index information, for specific portions of said at least one encrypted database which correspond to said search information;

decrypting at least one of said specific portions of said at least one encrypted database to produce corresponding decrypted information;

restricting use of at least one of said specific portions of said decrypted information to display only and preventing performance of said additional functions with respect to said specific portions;

determining at least one characteristic identifying said information decrypted by said decrypting step; and

storing said determined characteristic.

31. A method as in claim 30 wherein said method further includes selectively performing at least one of printing, storing, copying and communicating portions of the decrypted information restricted to display only by said restricting step.

32. A method as in any of claims 30 and 29 further including the following steps:

measuring the quantity of information decrypted by said decrypting step; and

storing said measured quantity information.

33. A method as in any of claims 30 and 29 wherein: said at least one database stored by said at least one storage medium is divided into plural discrete subdivisions;

said method further includes determining the subdivisions said selected portions are derived from; and

said storing step includes storing signals representing said determined subdivisions.

34. A method as in claim 30 wherein said displaying and preventing step includes the steps of:

further processing of said decrypted information by at least one of printing, storing, copying and communicating said decrypted information;

calculating a first cost in response to a first cost rate and also in response to the time duration over which said displaying step is performed;

calculating a second cost for performing said further processing step in response to a second cost rate different from said first rate and in response to at least one of (a) the quantity of information further processed, and (b) the time duration said information is further processed; and

storing said calculated first and second calculated costs.

35. A method as in claim 22 wherein said method 50 further includes the step of using said decrypted information and said metering step includes the step of determining the time duration over which at least one of (a) said searching step searches said information, (b) said reading step reads said information, (c) said decrypting step decrypts said read portion, and (d) said using step uses said information.

36. A method as in claim 28 wherein:

said searching step comprises the following steps:

(a) inputting search criteria determined at least in part in response to user input,

(b) referencing at least one index stored on said at least one storage medium and associated with said selected at least one database and searching said at least one index for index entries corresponding to said inputted search criteria,

(c) identifying database portions which correspond to said inputted search criteria, and

(d) selecting a subset of said identified database portions; and

said method further includes reading, from said at least one storage medium, said selected database portions.

37. A method as in claim 30 wherein:

said searching step comprises the following steps:

(a) inputting search criteria determined at least in partial response to user input,

(b) searching said index information for at least one index entry corresponding to said inputted search criteria,

(c) identifying at least one index entry which corresponds to said inputted search criteria, and

(d) selecting a subset of said at least one identified index entry; and

said reading and decrypting step includes reading from said storage means those at least one database portions corresponding to said selected subset of at least one identified index entry and decrypting said read at least one database portion.

38. A computer system comprising:

memory means for storing at least one characterized of database usage; and

signal processing means, operatively connected to said memory means and also operatively connected to at least one database including at least one encrypted textual component and associated index information, for performing the following functions:

searching said at least one database component in response to search criteria determined at least in part by user-input and retrieving at least one portion of said at least one textual database component which satisfies said search criteria;

decrypting said at least one retrieved portion of said at least one textual database component and producing a stream of signals corresponding to a decrypted version of said retrieved portions,

processing a portion of said signal stream in a first manner,

processing a portion of said signal stream in a second manner different from said first manner, and

storing in said memory means at least one of (a) the quantity of signals processed and (b) the time duration over which at least one of said first manner processing step and said second manner processing step is performed.

39. A method of accessing at least one database comprising the steps of:

storing text database information on a random access storage device;

searching for at least one portion of said stored at least one database based on search criteria determined at least in part by user input;

retrieving at least one portion located by said searching step;

using said retrieved at least one portion, determining a time duration corresponding to at least one of said searching and using steps;

metering information representing said determined time duration and storing said metered information in a further storage device different from said first-mentioned storage device; and

conveying said stored quantity-representing information to a location remote thereto.

40. A database access system comprising:

storage medium means for storing at least one encrypted textual database and at least one index associated with and corresponding to said at least one encrypted textual database;

memory means for storing information representing at least one database usage ceiling;

first input means operatively connected to said memory means for updating said stored at least one database usage ceiling;

further input means for generating database index search criteria in response to user input;

searching and retrieving means, operatively connected to said storage medium means and to said input means, for searching said at least one database, for referencing said at least one index based on said search criteria and for retrieving at least one portion of said encrypted at least one database from said storage medium means at least partially in response to said index referencing;

means operatively connected to said searching and retrieving means for decrypting said retrieved at least one encrypted database portion; and

control means operatively connected to said decrypting means and to said memory means for metering parameters of usage of information decrypted by said decrypting means, for comparing said metered usage with said at least one corresponding database usage ceiling stored in said memory means, and for preventing further decrypting by said decrypting means if said corresponding at least one database usage ceiling does not exceed said metered usage.

41. A database access system comprising:

storage medium means for storing at least one encrypted textual database and at least one index associated with and corresponding to said at least one encrypted textual database;

memory means for storing information representing a time of cessation of database usage;

real time clock means for providing an indication of real time;

first input means operatively connected to said memory means for updating said cessation time;

further input means for generating database index search criteria in response to user input;

searching and retrieving means, operatively connected to said storage medium means and to said input means for searching said at least one database, for referencing said at least one index based on said search criteria and for retrieving at least one portion of said at least one encrypted database from said storage medium means at least partially in response to said index referencing;

means operatively connected to said searching and retrieving means for decrypting said retrieved at least one encrypted database portion; and

control means operatively connected to said decrypting means, said real time clock means and said memory means for metering parameters of usage of information decrypted by said decrypting means, for comparing said real time with said stored cessation time, and for preventing further decrypting by said decrypting means if said stored cessation time is not later than said real time.

42. A secure data base access system comprising: at least one storage medium storing at least one encrypted textual database component and at least one index associated with said component;

input means for providing database index search criteria in response to user input;

searching means operatively connected to said at least one storage medium and to said input means for searching said database, including means for referencing at least one index based on said search criteria, for identifying portions of said encrypted at least one database in response to said index referencing, and for reading one of (a) all of said identified database portions, and (b) desired ones of said identified database portions from said at least one storage medium;

means operatively connected to said searching means, for decrypting at least one portion of said read encrypted information; and

control means operatively connected to said decrypting means for metering usage of information decrypted by said decrypting means and for preventing decryption of more than a predetermined percentage of said at least one database.

43. A system as in claim 42 wherein said control means measures the number of contiguous blocks of said textual information decrypted by said decrypting means and prevents said decrypting means from decrypting more than a certain number of said contiguous blocks.

44. A system as in claim 42 wherein said control means measures the time duration over which at least one of said searching means processes said information, and said information is decrypted, and wherein said metering means includes means for storing said measured time duration.

45. A system as in claim 42 wherein said metering means measures the duration of usage of said decrypted information.

46. A system as in claim 42 wherein said metering means includes means for storing said usage information.

47. A system as in claim 42 wherein:

said stored encrypted information has logically-related segments; and

said control means determines which logically-related segments said selected portions are associated with.

48. A system as in claim 42 wherein:

said medium stores plural textual databases; and

said control means meters usage of less than all of said databases.

49. A system as in claim 42 wherein said control means comprises:

a physically separable non-volatile memory device; and

monitoring means operatively connected to said decrypting means and also disengageably operatively connected to said memory device, for measuring the quantity of information decrypted by said decrypting means and for storing said measured quantity in said memory device.

50. A system as in claim 49 wherein said monitoring means disables said decrypting means from operating unless said memory device is operatively engaged thereto.

51. A system as in claim 49 wherein said control means comprises:

memory means for storing usage information;

monitoring means, operatively connected to said decrypting means and also operatively connected to said memory means, for metering at least one of (a) the quantity of information decrypted by said

decrypting means, (b) an identification of a subset of information stored on said medium containing said identified portions, and (c) duration of at least one of searching, identifying, decrypting, reading and using of said database portions, for generating signals indicating the result of said metering, and storing said generated signals in said memory means; and

means operatively connected to said decrypting means and to said memory means for preventing said decrypting means from decrypting information whenever said metered indicating signals are not successfully stored in said memory means.

52. A system as in claim 42 wherein said control and communicating means includes;

a memory; and
monitoring means, operatively connected to said decrypting means and to a communications network, for monitoring the quantity of at least one of: (a) information decrypted by said decrypting means and (b) information identified by said searching means, for controlling said signal communicating means to communicate an indication of said monitoring to a billing facility over said communications network.

53. A system as in claim 42 wherein said monitoring means also determines identifying characteristics of at least one of (a) said decrypted portions (b) of said identified portions and controls said signal communicating means to communicate said identifying characteristics to said billing facility.

54. A system as in claim 42 wherein:
said at least one storage medium also stores unencrypted index information correlating unencrypted search information with portions of said at least one encrypted database.

55. A system as in claim 42 wherein:
said at least one storage medium also stores encrypted index information correlating search information

with portions of said encrypted at least one database.

56. A system as in claim 42 further including:

a first memory means, operatively connected to said decrypting means, for storing said decrypted information; and

a second memory means, operatively connected to said metering and communicating means and different from said first memory means, for storing said metered usage.

57. A secure database access system comprising:

at least one storage medium storing at least one textual database in encrypted form, said at least one database including at least one collection of textual information, said at least one storage medium also storing index information, said index information correlating portions of said at least one encrypted database with search information;

at least one host signal processor, operatively connected to said at least one storage medium, said at least one processor preprogrammed so as to: (a) accept search criteria in response to user input thereto, (b) search said index information, (c) identify, in accordance with said search of index information, the portions of said at least one encrypted database which satisfy said search criteria, and (d) reach at least one of said identified encrypted database portions from said at least one storage medium;

means for decrypting at least one portion of said at least one encrypted database to produce corresponding decrypted information; and

decoder control logic means, operatively coupled to said host processor and said decrypting means and adapted for operatively connecting to a telecommunications network, for metering the usage of information decrypted by said decrypting means, wherein said decoder control logic means includes means for telecommunicating said metered usage over said telecommunications network to a remote location.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 4,977,594

Page 1 of 2

DATED : December 11, 1990

INVENTOR(S): Victor H. SHEAR

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

- Col. 24, line 14, "claim 1" should read --claim 8--.
- Col. 25, line 59, "ready" should read --read by--.
- Col. 28, line 17, delete "reading and".
- Col. 28, line 52, after "information" insert --in encrypted form--.
- Col. 28, line 55, after "one" insert --encrypted--.
- Col. 28, line 57, delete "at least one" and insert --an encrypted database--.
- Col. 28, line 59, before "using" insert --decrypting and--.
- Col. 28, line 59, delete "at least one".
- Col. 28, lines 60 and 61, delete in their entirety.
- Col. 28, line 62, after "metering" insert --the quantity of--.
- Col. 28, line 62, delete "representing said determined".
- Col. 28, line 63, delete "time duration" and insert --used by said decrypting and using step--.
- Col. 28, line 63, delete "information" and insert --quantity--.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 4,977,594

Page 2 of 2

DATED : December 11, 1990

INVENTOR(S) : Victor H. Shear

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Col. 32, line 27, "reach" should be --read--.

Signed and Sealed this
Eighteenth Day of May, 1993

Attest:



MICHAEL K. KIRK

Attesting Officer

Acting Commissioner of Patents and Trademarks

EXHIBIT H



United States Patent [19]

[11] **Patent Number:** **6,029,065**

Shah

[45] **Date of Patent:** Feb. 22, 2000

- [54] REMOTE FEATURE CODE PROGRAMMING
-
- FOR MOBILE STATIONS

5,794,156	8/1998	Alanara	455/517
5,842,124	11/1998	Kenagy et al.	455/418

[75] Inventor: **Bharat Shah**, San Diego, Calif.

[73] Assignee: **Nokia Mobile Phones, Ltd., Espoo, Finland**

[21] Appl. No.: 08/841,850

[22] Filed: May 5, 1997

[51] Int. Cl.⁷ H04Q 7/20

[52] **U.S. Cl.** 455/414; 455/415; 455/418;
455/419; 379/201; 379/211; 370/259; 370/271

[58] **Field of Search** 455/414, 415,
455/416, 417, 418, 419, 432, 435, 550;
370/271, 259, 465; 379/201, 207, 211,
212, 214

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,159,625	10/1992	Zicker	379/59
5,216,703	6/1993	Roy	379/59
5,297,191	3/1994	Gerszberg	455/419
5,297,192	3/1994	Gerszberg	455/419
5,301,232	4/1994	Mulford	380/21
5,343,494	8/1994	Averst et al.	373/15
5,381,138	1/1995	Stair et al.	340,825/44
5,404,355	4/1995	Raith	370/311
5,418,837	5/1995	Johansson	455/588
5,425,077	5/1995	Isol	379/58
5,459,774	10/1995	Breden	379/58
5,478,419	1/1996	Antoniou	455/550
5,533,019	7/1996	Jayapalan	370/382
5,577,100	11/1996	McGregor et al.	379/58
5,577,103	11/1996	Foti	455/412
5,590,398	12/1996	Mathews	455/331
5,794,142	8/1998	Vanilla et al.	455/419

OTHER PUBLICATIONS

Vertical Service Code Assignment Guidelines, Doc. No. Inc 96-0802-015, Industry Carriers Compatibility Forum (ICCF), Revised Aug. 2, 1996.

Primary Examiner—Wellington Chin

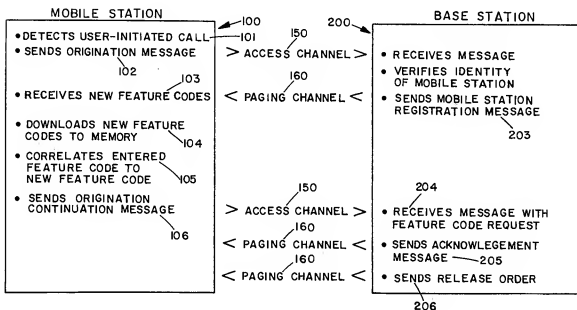
Assistant Examiner—Isaak R. Jarm

Attorney, Agent, or Firm—Brown Martin Haller & McClain, LLP

[57] ABSTRACT

The base station of a wireless communication network determines what features a mobile station will support, then downloads information to the mobile station which will notify the mobile station of which network features are available and how they may be accessed in the local network. Specifically, the base station provides the features codes that are required to access the network features. The base station may also inquire into what features the mobile station supports. Each of these communications may take place over the combination of the Paging Channel/Access Channel, collectively, the Control Channel, or the Traffic Channel. With the downloaded information, the mobile station user may select a desired feature using the method to which he or she is accustomed, i.e., either by selecting a menu location or by entering a familiar sequence of key-strokes. The mobile station's internal processor converts the entered values into the feature codes corresponding to the selected features within the network. The process of downloading the feature code information does not require the user's intervention, such that any conversion required from the user's familiar feature access process is transparent to the user.

29 Claims, 3 Drawing Sheets



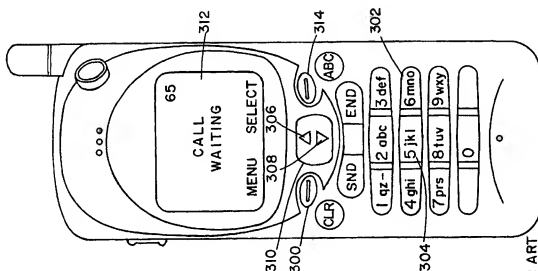


FIG. 2 PRIOR ART

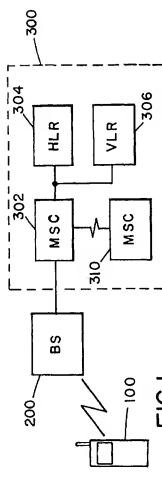


FIG. 3 PRIOR ART

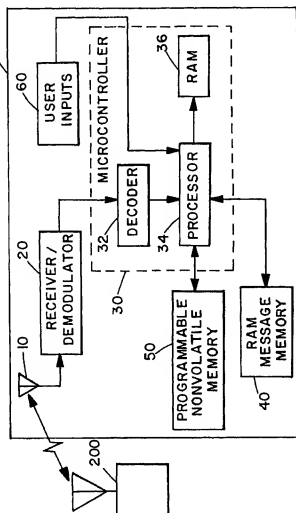


FIG. 1 PRIOR ART

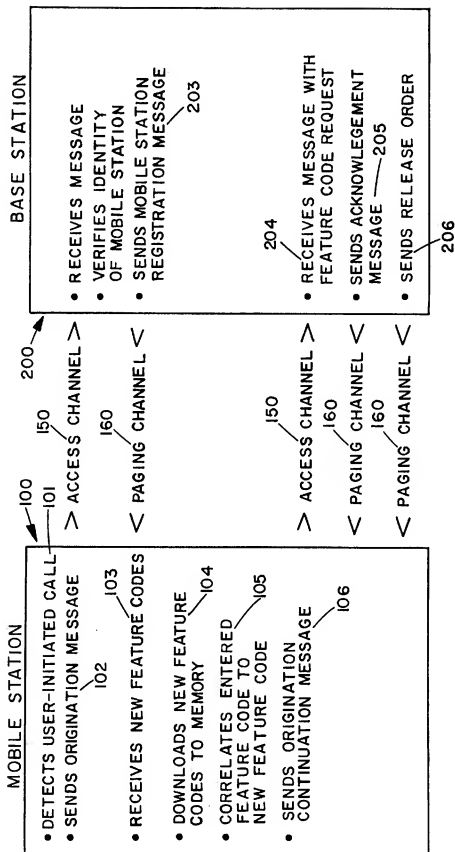


FIG. 4

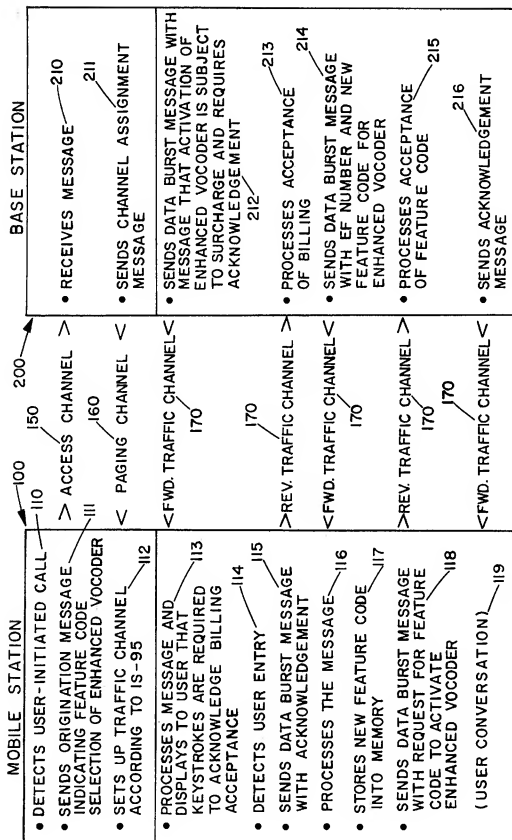


FIG. 5

REMOTE FEATURE CODE PROGRAMMING FOR MOBILE STATIONS

FIELD OF THE INVENTION

This invention relates generally to programming of network feature codes in mobile stations, and more specifically for providing a user-transparent conversion of one or more of a set of network feature codes in a mobile station to facilitate use of network features which must be accessed using a different set of feature codes.

BACKGROUND OF THE INVENTION

Telephone service carriers, including both mobile cellular and PCS networks, provide a number of different features which may be used by a network subscriber at the subscriber's discretion. A non-exhaustive list of features which may be provided include Call Forwarding, Call Waiting, Calling Number Identification, Automatic Callback, Conference Calling, Message Waiting Notification, Call Encryption, Selective Call Acceptance, Voice Mail, Enhanced Vocoder, and Cost of Call Notification. Activation or deactivation of such features typically involves the completion of a sequence of keystrokes on the keypad to enter the "feature code", also known as a "vertical service code" or "VSC", for selecting or de-selecting the desired feature. The keystrokes may involve the entry of a numeric sequence preceded by the star key (*) and/or followed by the pound key (#), e.g., "**66#", or may involve the selection of a dedicated menu by pressing a "Menu" key followed by a one- or two-digit number, one or more soft keys, or by scrolling the menu screen. Even when a mobile unit has menu capability, the feature code value specific to the home network is commonly a numeric sequence as above, with selection of the "Menu" key being sent as a star key to indicate a feature code. In this case, the feature code value is programmed upon activation of the feature to correspond to a particular menu location, and the actual feature code value is unknown to the user.

While the network subscriber may become thoroughly familiar with the procedure for access feature codes within his or her home network and mobile phone, several obstacles to the optimal usage of such features exist as a result of the lack of standardization within the communications industry. These obstacles may appear because, for example, the feature code for any given feature may be different from one network to another, not all networks provide all features, and there is a large number of different types of telephones, many of which may not support such features, or which have their own pre-determined menu-based sequence for selecting feature codes. While these issues may pose little practical difficulty for wire-based network subscribers, mobile phone users frequently travel between coverage areas of different network service providers and, thus, are likely to experience the loss of, or inability to control, a desired feature. Under current practices, the only way that a mobile user can determine whether his or her feature code matches the corresponding code for the visited network is to try it. If the entered value matches, an acknowledging tone or message will be received at the mobile unit. If the code does not match, a different tone or message will indicate an unsuccessful entry. While little may be done if a mobile user has entered an area covered by a visited network that simply does not offer a desired feature, the typical user does not want to go to the time and trouble if, even when the visited network provides the feature, the only way to control the feature is to request the assistance of that network's cus-

tomers service representative. In addition to the problem of dealing with different feature codes used in visited networks, even if the mobile user is not traveling between networks, advances in technology and increased competition between service providers is resulting in the availability of new network features which must be enabled in the mobile stations of existing subscribers within the mobile's home network.

FIG. 1 provides a diagram of the functional entities within a network to illustrate how feature codes are handled in a conventional wireless communications network. Generally, the network consists of the Mobile Station 100, the Base Station 200 and the switching system 300 (designated by dashed lines). Within switching system 300 is the Mobile Switching Center (MSC) 302, for the first network, Home Location Register (HLR) 304 and Visitor Location Register (VLR) 306. MSC 310 is a second Mobile Switching Center corresponding to the home network for Mobile Station 100, which is connected to MSC 302 via conventional wire-based connection for communicating information regarding MS 100 to the visited network. Under current feature code implementation procedures, once MSC 302 receives a "*" or "Menu" input ("Menu" will be seen by the MSC as a star) from MS 100 (via BS 200), it transfers the information to HLR 304, which stores and manages subscriptions, and contains permanent subscriber information. If MS 100 is operating within its home network, HLR 304 will recognize the feature code entered by the mobile user and provide the requested response. If MS 100 is outside of its home network, MSC 302 will transfer information about the mobile station to VLR 304, which contains temporary information needed by MSC 302 to serve visiting mobile stations. If a star key ("*") is pressed followed by two digits to activate a feature code by a visiting mobile station, the message will still be processed through HLR 304. Using procedures established under the IS-41 standard for inter-network communications, MSC 310 is contacted by MSC 302 once it is determined that MS 100 is outside of its home network to provide the information necessary for MSC 302 to communicate with MS 100. Since HLR 304 processes feature codes under current procedures, and since it contains only information regarding the system's own subscribers, it is unable to process feature codes from visiting mobile stations if the requested feature code value differs from its own value. Attempts by a visiting mobile station to access a feature code with a different value, or a feature not supported by the visited system, will be met with a busy signal, or some other indication that the requested feature is unavailable.

Industry efforts are being made to standardize certain feature codes and the requirements for activating them under Telecommunications Industry Association/Electronics Industries Association Interim Standard 52-A (TIA/EIA/IS-52-A), and/or INC 96-0802-015 ("Vertical Service Code Assignment Guidelines"), incorporated herein by reference, which will be of assistance with new networks and new mobile units once the standardization is completed. Delays in implementation of the new standardized feature codes in existing networks are likely to be on the order of two or three years from Final Closure of the issue, which occurred in August 1996.

The newer standards provide for a shift of the processing of feature codes to the Mobile Switching Center, which, while having the advantage of removing the previous limitation created by handling the feature codes in the Home Location Register, still requires standardization of the feature codes amongst all mobile phone manufacturers in order to make sure that every mobile station uses the feature code

recognized by the MSCs. Because many mobile phone manufacturers have developed their own systems and preferences, it is unlikely that a consensus could be achieved for adopting a single menu standard or a single numeric sequence for accessing feature codes from every mobile phone, regardless of manufacturer. Further, new features are certain to be added in the future, which will need to be implemented in existing networks and subscriber mobile units. Accordingly, the need remains for means to enable mobile users to readily access network features across multiple networks without requiring the user to learn additional feature codes.

SUMMARY OF THE INVENTION

It is an advantage of the present invention to provide means for allowing a mobile user to access network features without requiring knowledge of the network's specific feature codes.

It is another advantage of the present invention to provide means for programming a mobile station with information to provide a user-transparent conversion of a first set of network feature codes to a different, second set of feature codes which are recognized by the mobile station.

In an exemplary embodiment, the method for remote feature code programming first registers a visiting mobile station using the standard registration procedures provided for in the established industry standards to notify the visited network base station of the mobile station's presence. According to these standards, which include the TIA/EIA/IS-95 and ANSI-J-008 standards, when a mobile user visits a network, registration procedures are used to enable the visited network to identify the mobile unit network connection and paging purposes, and the mobile station's home network, for billing purposes. Once registered, the base station will download information to the mobile station which will notify the mobile station of which network features are available and how they may be accessed in the local network. The base station may also inquire into what features the mobile station supports. Each of these communications may take place over the combination of the Paging Channel/Access Channel, collectively, the Control Channel, or the Traffic Channel. With the downloaded information, the mobile station user may select a desired feature using the method to which he or she is accustomed, i.e., either by selecting a menu location or by entering a familiar sequence of keystrokes. The mobile station's internal processor converts the entered values into the feature codes corresponding to the selected features within the network. The process of downloading the feature code information does not require the user's intervention, such that any conversion required from the user's familiar feature access process is transparent to the user.

The downloading step for providing a new set of feature codes may be included in any of a number of different established messages. For example, using the Control Channel, the base station may include the feature code data in either the Mobile Station Registered Message, when the mobile station attempts to originate a call after registration, or in some other forward channel message, such as a Page Message. The process may also occur over the Traffic Channel, for example, in a Data Burst Message, or can be done using a combination of messages communicated over the Control and Traffic Channels, such as in the Over-the-Air Parameter Administration (OTAPA) method disclosed in co-pending application Ser. No. 08/837,970, filed Apr. 15, 1997, of the present inventor.

The downloaded information may contain a list of network features that are supported by the visited network and how those features may be accessed. This access information may be stored in the mobile station's memory as Reverse Channel Information Records, or in some other memory location to which the mobile station's processor may refer to translate entries from the mobile station's menu, or other known feature code keying sequences, into the system-specific feature codes for the visited network. When the mobile user selects a feature to activate or deactivate, the sequence of keystrokes with which he or she is familiar, or which is selected via the menu display of the mobile unit, is entered. The information for the selected feature which is stored in the Reverse Channel Information Records is then transmitted to the base station with an appropriate reverse channel message, such as a Page Response Message, Status Message, or Flash with Information Message, along with a request for acknowledgment, when the mobile user wishes to activate or deactivate the feature. The base station responds indicating activation or deactivation of the selected feature. If the visited network does not support a particular feature, the base station will respond with an indication of that fact, causing a busy signal or some other tone to be heard by the user. If the mobile unit has a display screen, information the phrase "feature not available" may be displayed following entry of the keystrokes for that feature.

When new features become available within a network, even within the mobile station's home network, the network base station can download the information required to access the new feature using the same procedure. Similarly, if the home network changes its feature code values, for example, if it adopts a standardized set of feature code values, the home network will be able to provide this information to the mobile station without requiring the mobile user to relearn an entire set of feature codes. If the mobile unit does not have a display, external notification of the new feature should be provided to the subscriber, e.g., by mail. If the mobile unit has a display screen, the information downloaded can include a message for display to the user to notify the user of the new feature and how it may be accessed. For mobile phones with menu displays, the new feature may be added to the menu, and may be accessed by stepping or scrolling through the menu.

Downloading of the local feature codes does not require access of the mobile unit's protected memory, and the information can be exchanged as part of standard registration procedures, over the Control Channel, or over the Traffic Channel. This procedure may be used regardless of whether the mobile unit is located in its home network or in a visited network. Since the local feature codes may be changed frequently if the mobile user travels often, and since the information can be downloaded/refreshed during every activation procedure or paging sequence, if necessary, the local network's feature codes may be stored in the phone's temporary memory, much like a list of recent calls.

Where a feature is subject to a subscription charge and requires provisioning, for example, a high quality vocoder option, information regarding support of such a feature, selection of the appropriate feature code, and acceptance of billing terms may preferably be communicated on the Traffic Channel due to that fact that a number of messages and responses may need to occur for the transaction. During a combination of communications over the Control and Traffic Channels, the transactions can occur for notifying the user of a surcharge, responding with acceptance of the surcharge, and activation of the feature if accepted by the user.

BRIEF DESCRIPTION OF THE DRAWINGS

Understanding of the present invention will be facilitated by consideration of the following detailed description of preferred embodiments of the present invention taken in conjunction with the accompanying drawings, in which like numerals refer to like parts, and in which:

FIG. 1 is a block diagram of an exemplary prior art communications network including a mobile station, a base station and a switching system;

FIG. 2 is a block diagram of an exemplary mobile phone receiver;

FIG. 3 is a diagrammatic view of a mobile unit handset with a menu display;

FIG. 4 is a diagram of an exemplary call flow for changing a Feature Code List according to one embodiment of the present invention; and

FIG. 5 is a diagram of an exemplary call flow for changing a feature code for a provisioned feature according to another embodiment of the invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The following detailed description utilizes a number of acronyms which are generally well known in the art. While definitions are typically provided with the first instance of each acronym, for convenience, Table 1 below provides a list of the acronyms and abbreviations used herein along with their respective definitions.

TABLE 1

ACRONYM	DEFINITION
ANSI	American National Standards Institute
BS	Base Station
CDMA	Code Division Multiple Access
EF	Extended Feature
EFCC	Extended Feature Change Code
EIA	Electronics Industries Association
ESN	Electronic Serial Number
HLR	Home Location Register
INC	Industry Numbering Committee
IMSI	International Mobile Station Identity
IS	Interim Standard
MIN	Mobile Identification Number
MS	Mobile Station
MSC	Mobile Switching Center
NID	Network Identification
OTAPA	Over-the-Air Parameter Administration
OTASP	Over-the-Air Service Provisioning
RAM	Random Access Memory
SID	System Identification
SMS	Short Message Service
TIA	Telecommunications Industry Association
TSB	Technical Service Bulletin
VLR	Visitor Location Register
VSC	Vertical Service Code

It should be noted the font variations within the specification and claims, particularly italicized text and text in all capital letters, reflect the formats established according to the various standards which are applicable to wireless communications, e.g., IS-95.

Referring to FIG. 2, a conventional mobile phone receiver, which is generally designated by reference numeral 100, typically comprises an antenna 10 for receiving call signals. The received signals are provided to a receiver and a demodulator circuit 20 and the resulting digital signal is fed to a microcontroller 30. Microcontroller 30 comprises a decoder 32, a processor 34, and an internal RAM 36. The

digital signal is decoded by the decoder 32 and processed by processor 34, which reads from or writes to internal RAM 36. Memory storage for optional features, messages, and other data is provided by RAM 40. The user can enter data into processor 34 via user input circuitry 60.

A non-volatile memory 50 is coupled to processor 34 for storage of information necessary for the operation of mobile phone receiver 100. The memory can be an electrically erasable programmable read only memory (EEPROM), a battery backed up memory device, or a similar memory device which retains information even when power is not applied to the mobile phone. Processor 34 accesses information such as options for various features from non-volatile memory 50 during operation, and can alter information in the memory 50 by reprogramming in accordance with the present invention.

The following detailed description specifically refers to the protocol and procedures for operating a CDMA network conforming with the TIA/EIA-IS-95 standard (which is incorporated herein by reference), however, it will be apparent to those skilled in the art that by substituting the corresponding protocol and processes for various analog or PCS network architectures, the inventive feature code programming method may be similarly implemented in other wireless communications networks.

Stored within the programmable non-volatile memory 50 of mobile phone receiver 100 is the phone's identifying information, including its mobile identification number (MIN for analog; IMSI for CDMA), electronic serial number (ESN), network identification (NID), system identification (SID), home registration indicator (HOME₁₃ REG), along with other information which will allow the network base station 200 to locate, page, and determine the characteristics and capabilities of the mobile station. According to IS-95 procedures, certain of this information must be changed within the mobile station's memory in order for the mobile station to operate within a visited network.

Whenever the mobile station enters the visited network, registration procedures are initiated. As is known in the art, the manner of initiation of registration procedures depends upon how the mobile station is set-up, and may be autonomously initiated by time, distance, zone, powering up and/or powering down the unit while the mobile unit is roaming. Autonomous registration procedures may take place when the mobile unit is active, while on the Traffic Channel, or when it is in the Mobile Station Idle State, while the mobile unit monitors the Paging Channel, and does not require the user's intervention. Other registration procedures occur when the mobile station changes its stored parameters or when it enters a new system (Parameter-change registration), or when it receives messages from the base station over a Paging Channel or Traffic Channel. Generally, the autonomous registration procedures are directed toward preparing the mobile station for communicating with the base station so that registration can be completed once the mobile station is activated for communicating on a Traffic Channel.

Continuing the registration process as provided under standardized procedures, a network base station sends system overhead messages on the Paging Channel to provide the mobile station with the information it needs to operate with the base station. Included within the overhead messages is the identifying and operating information for the base station. A mobile station in the Mobile Station Idle State monitors the Paging Channels in accordance with standardized procedures. Upon receipt of a Page Message

containing the identifying and operating information for the base station, the mobile station compares the configuration message sequence number to the corresponding number stored in its memory. If the comparison results in a match, the system parameters are the same as those already stored, and the mobile station ignores the message. If a mismatch results, the mobile station enters the Update Overhead Information Substate and stores the appropriate parameters as Reverse Channel Information Records. Detection of a mismatch at this point may be used to trigger the procedure for downloading a new set of feature codes from the base station to the mobile station. The new set of feature codes may be included in this message in one embodiment of the programming method. The mobile station transmits a Page Response Message to confirm the downloading of the updated parameters and to notify the base station that the mobile station subscribes to certain network features.

In another embodiment, rather than downloading the feature codes over the Paging Channel, the procedure continues. Still on the Paging Channel, the base station sends a Channel Assignment Message directing the mobile station to move to an assigned Traffic Channel. The mobile station processes the Channel Assignment Message and sets up the designated Traffic Channel according to IS-95 procedures. The base station transmits a Status Request Message inquiring into the features supported by the mobile station, to which the mobile station responds with a Status Response Message to identify the supported features. (Note that this exchange may occur even if the mobile station is in the Conversation Substate.)

During any one of the above Forward Channel communications, the base station can include a message containing the corresponding feature codes that are to be used in the visited network. These new feature codes are stored in the mobile station's memory as additional Reverse Channel Information Records. The mobile station's processor will then transmit these new feature codes in place of the previously-programmed feature codes whenever the user wishes to access a particular feature. Thereafter, when the mobile user attempts to activate or deactivate a particular network feature, entry of the keystroke sequence to which the user is accustomed will automatically cause the mobile station to transmit the appropriate system-specific feature code to the base station.

Alternatively, rather than requiring an additional inquiry into the supported feature codes, since the base station has already ascertained that the mobile station is likely to have a different set of feature codes based upon the mismatch of other system and network identities during registration, the new feature code information may be automatically transmitted from the base station along with the Mobile Station Registered Message. This message, which includes the parameters for communicating with the base station, will store the new system and network information into the mobile's memory.

In an embodiment in which the mobile user subscribes to a network feature for which a subscription fee is charged, such as enhanced vocoder capability, voice mail, voice dialing, or conference calling, the feature must be provisioned, when initially subscribed to, when the mobile user wishes to utilize the feature outside of his or her home network, and/or when the subscribed features are changed. Access to the feature, hereinafter referred to as an "extended feature", may be provided using the over-the-air programming protocol and procedures which support the Over-The-Air Service Provisioning (OTASP) feature in accordance with established industry standards (TIA/EIA/IS-683) in a

procedure for Over-The-Air Parameter Administration (OTAPA). A difference between the OTAPA and OTASP procedures is that initiation of the OTAPA procedure does not require the intervention of the mobile user.

The mobile phone is pre-programmed with a service option for changing or adding extended subscriber features, which includes assignment of an Extended Feature (EF) number. The mobile phone will also have one or more extended features change codes (EFCCs) in its memory. The network, whether it is the mobile's home network or a visited network, possesses means for determining whether a mobile phone is OTAPA capable. Note that the visited network may establish OTAPA support for a particular mobile station using IS-41 communications with the home network, however, protocol for the transfer of such information will need to be added to the IS-41 standard.

In the OTAPA procedure, the network base station sends a General Page Message to the mobile phone using the EF number. After first verifying its identity using the standardized Authentication process, if the mobile phone has OTAPA capability, it responds with a Page Response Message, indicating support for the EF by sending the EF number. If the mobile station does not support the option, the response will indicate that the option is not available. Once the presence of the option is confirmed, the base station transmits a Channel Assignment Message, telling the mobile station to proceed to the Traffic Channel. In order to prevent unauthorized access to the mobile user's billing records, it may be desirable to use the Signaling Message Encryption (SME).

Once the mobile station is on the Traffic Channel, an OTASP Data Message is sent that an additional fee is charged for the use of the feature and requesting acknowledgment of acceptance. If accepted, a second OTASP Data Message is sent containing a Extended Feature Change Code (EFCC). If the EFCC matches the EFCC for the mobile station, it is verified by the mobile unit, after which it may be used to unlock the mobile station, update the feature code(s) and store the updated feature code(s) into the phone's memory. After verification of the programmed data in accordance with OTASP processing, the process is terminated. If the user refuses the additional billing, no downloading will occur. A number of different EFCCs may be used for different feature codes so that the user may elect the feature codes individually to avoid being billed for access to all possible optional extended features when only one is desired.

As specified in the IS-683 standards, delivery of OTASP Data Messages does not require sequential delivery of messages by the layer 2 protocol because OTASP procedures ensure that only one OTASP Data Message is outstanding at any time. Therefore, a Data Burst Message (OTASP BURST_TYPE) may be used in accordance with TIA/EIA/58, "Administration of Parameter Value Assignments for TIA/EIA Wideband Spectrum Standards". Data Burst Messages may also be used on the Control Channel (Paging/Access Channel) per IS-95 specifications.

Using a similar procedure, a visited network will need to confirm that the visiting mobile user is willing to pay for use of a feature within the visited network when the mobile user can use the same feature in his or her home network without additional charge. For example, call forwarding may be provided at no extra charge by some network providers, while others add a surcharge for the service. In this case, the visited network base station will send a message to the mobile station that a selected feature can be accessed only

with payment of a toll by causing a message to be displayed on the display of the mobile station such as "Toll for Feature". The base station will then wait for confirmation from the mobile station before downloading the feature code which will allow activation of the feature.

It should be noted that the system-specific feature code information need not be included within system overhead information, nor is it limited to inclusion in a System Parameter Message, Mobile Station Registered Message, OTASP Data Message, or Data Burst Message. These messages are exemplary only. The system-specific feature code information can be included in any message, whether transmitted over a Paging Channel or a Traffic Channel, which is capable of initiating a downloading operation to the mobile station when a discrepancy exists between certain system operating parameters, in this case, the feature codes, and the access-enabling parameters stored within the mobile station. Whichever message type is used, an appropriate header or data field will need to be provided to designate the nature of the message as relating to feature code updating. As in the case of any new feature in standardized systems and data formats, addition of new system overhead information may require changes to existing IS-41 (Network Specification) specified messages, or other new messages may need to be defined by the standards organizations. Initiating such changes is an administrative task that will be readily apparent to those in the art.

To provide an example of the invisible nature of the feature code translation, FIG. 3 illustrates a Nokia® Model 2120 mobile phone. With the mobile phone initially in the Idle State, the "Menu" soft key 300 is pressed followed by numerical keys "6" 302 and "5" 304 to select the "Call Waiting" feature. Alternatively, the user can press the "Menu" soft key 300, then step forward or backward through the list of feature by pressing the "up arrow" 306 or "down arrow" 308 on key 310.

Once the desired function is displayed on display screen 312, the user then presses the "Select" soft key 314. To activate the Call Waiting feature, the user presses "up arrow" 306. To deactivate Call Waiting, the user presses "down arrow" 308.

Due to the differences between mobile handsets from different manufacturers, keys which perform a similar function may be labeled differently. For example, some manufacturers will call their equivalent of the Nokia® "Menu" key a "Feature" key. Other manufacturers may use different combinations of soft keys and hard keys. With the present method of remotely programming feature codes, the mobile user can utilize his or her preferred format (as selected based upon the chosen make of mobile handset), use the feature code selection method to which he or she is accustomed, and still be able to access the desired feature regardless of how dissimilarly a visited network has its feature codes set up.

Referring briefly to FIG. 2, the preceding keystroke sequence comprises a user input via user input 60 to processor 34. Using the system-specific feature code information stored in memory (either non-volatile memory 50 or RAM memory 40), processor 34 converts the selected feature into the appropriate feature code for the network which is then transmitted to the base station for implementation (activation or deactivation).

For mobile station handsets which do not have menu display capability, features are typically selected by a pressing a keystroke sequence designated by the mobile's home network. For example, if "66#" ("Send") is designated by the home network service provider for activating Call

Waiting, and "67#" ("Send") for deactivating Call Waiting, the mobile user will press the four keys in the proper sequence. To continue the example, the visited network's feature codes for Call Waiting are "55#" and "56#" for activation and deactivation, respectively. Following the downloading of the correct system parameters, including the system-specific feature codes of the visited network, the mobile station's processor will cause the mobile station to transmit a "55#", even when the user has pressed "66#".

Where a feature code is entered preceding a phone number, for example, for Call Forwarding, it may be desirable to limit the translation of feature codes to a specific number of digits following the "##", so that the subsequent phone number is not also inadvertently converted. Again using the situation where the feature codes are selected by a two digit number, in the home network, Call Forwarding may be activated by entering "44555-4444#" ("Send"), where 555-4444 is the phone number. In the visited network, the feature code to activate Call Forwarding may be "55". To avoid the possibility of conversion of the phone number as well as the feature code, the processor would look only at the first two digits entered following the star key.

In a second embodiment of the method for remote programming of feature codes, the mobile station completes the registration procedure in the visited network, making it capable of communicating with the local base station. Rather than automatically downloading the local feature codes into the visiting mobile station's memory at the time of, or in response to the initial activation of the mobile station within the network, the base station waits until it receives an Origination Message from the mobile station over the Access Channel for a mobile-initiated call which includes an indication, i.e., the star key ("*"), that the mobile user wishes to access a feature, e.g., Call Forwarding. If the particular feature code value is not recognized by the base station, it may respond over the Paging Channel, or on the Traffic Channel after providing a channel assignment, with information containing the local feature code values for downloading into the mobile station's memory. The processor within the mobile station determines the correct local feature code and responds with an Origination Continuation Message to activate the Call Forwarding feature by re-sending the message using the correct feature code values for the visited network. In this embodiment, information regarding local feature code values is provided only when access to a feature is requested by the mobile user. Because the downloading of the feature code information is triggered by the request for access to a feature, the mobile user need not perform any additional steps beyond that to which he or she is already accustomed, with the conversion occurring internally within the mobile station's processor.

FIG. 4 provides an exemplary flow diagram for accessing a network-specific features code using a visiting mobile station in a visited network. The following assumptions apply:

- 1) The Mobile Station and Base Station each supports the Call Waiting feature without surcharge;
- 2) The Mobile Station's home network utilizes a different feature code for Call Waiting than is used by the visited network in which the Base Station is located; and
- 3) The Mobile Station has already performed the initial steps for registration in the visited network, including updating overhead information.

Using the example provided in FIG. 3, the user wishes to activate Call Waiting and enters "Menu", "6", "5", and

"Send". Mobile Station 100 detects a user initiated call 101 which includes selection of a feature code (by depressing the "Menu" key) and sends an Origination Message 101 over Access Channel 150 to the visited network Base Station 200. Included in Origination Message 101 is the indication that a feature code is selected. Base Station 200 receives the message (Step 201), verifies the identity of the mobile station (step 202) and responds with a Mobile Station Registration Message 203 on the Paging Channel 160 which includes the new system-specific feature code values for the visited network. Mobile Station 100 processes the received new feature codes (step 103) and downloads the information into its memory (step 104). Processor 34 correlates the entered value ("Menu", "6", "5") with the system-specific feature code (*55) that was previously downloaded (step 105) and sends an Origination Continuation Message 106 which acknowledges the received information and includes the translated feature code value (*55). Base Station 200 received the message (step 204) and responds with an Acknowledgment Message 205, acknowledging activation of the Call Waiting feature. The call is then released (step 206).

In this case, the mobile user did not have to enter any different feature codes since the Mobile Station's processor automatically and transparently converted the feature code to the correct value for communicating with the visited network.

In a second exemplary call flow shown in FIG. 5, the feature to be selected is one which requires provisioning. The following assumptions apply:

- 1) The Mobile Station and Base Station each supports the Enhanced Vocoder feature;
- 2) The Mobile Station's is initially programmed to utilize a different feature code for Enhanced Vocoder than is used by the Base Station; and
- 3) The Mobile Station is already registered with the Base Station.

In FIG. 5, the Mobile Station is, again, generally designated as 100 and the Base Station as 200. Mobile Station 100 detects a user initiated call 110 which includes selection of a feature code (by depressing the star key) and sends an Origination Message 111 over Access Channel 150 to the visited network Base Station 200. Included in Origination Message 111 is the indication that a feature code is selected. Base Station 200 received the message (step 210) and sets up a Traffic Channel, then sends a Channel Assignment Message 211 on the Paging Channel 160. Mobile Station 100 processes the Channel Assignment Message 211 and sets up the designated Traffic Channel 170 according to IS-95 procedures (step 112).

Base Station 200 sends a Data Burst Message 212 using OTASP BURST-TYPE fields which include information that the selected feature code is subject to a surcharge and a request for acceptance of the surcharge. Mobile Station 100 processes Data Burst Message 212 and indicates to the user that acknowledgment of acceptance is required (step 113). The user enters his or her acceptance by pressing the appropriate key (step 114), which is transmitted over the Reverse Traffic Channel to Base Station 200 as Data Burst Message 115. Base Station 200 processes the acceptance of billing (step 213) and sends a second Data Burst Message 214 including the EF number and the feature code for the Enhanced Vocoder feature. Mobile Station 100 processes Message 214 (step 117) and responds with Data Burst Message 118 requesting activation of the Enhanced Vocoder feature. Base Station 200 receives and processes the request

(step 215), then sends Data Burst Message 216 with an indication that the feature is activated. The mobile user then proceeds with his or her conversation (step 119).

In addition to its application to allowing a mobile station to access network features when visiting other networks, the procedure for remotely programming feature codes in a mobile station can also be used in the mobile station's home network when new features or made available, or if the network adopts a feature code standard which differs from its previously designated feature codes. Further, the remote programming feature is not limited to mobile telephones and voice communications devices, but may also include pagers and data communications devices, which also utilize network features and, thus, are subject to loss of features when traveling outside of a home network, or which will require updating when new network features become available.

Other embodiments and modifications of the present invention will occur readily to those skilled in the art in view of these teachings. Therefore, this invention is to be limited only by the following claims, which include other embodiments and modifications when viewed in conjunction with the above specification and accompanying drawings.

I claim:

1. A method for accessing a plurality of network features enabled by a local set of feature codes using a mobile station having an original set of feature codes, the local set of feature codes being at least partially incompatible with the original set of feature codes, the method comprising:

communicating between a network base station and the mobile station to register the mobile station with the network base station;

transmitting the local set of feature codes from the base station to the mobile station;

comparing the local set of feature codes with the original set of feature codes;

if the local set of feature codes is at least partially incompatible with the original set of feature codes, storing the local set of feature codes within a memory of the mobile station;

selecting an original feature code from the original set of feature codes corresponding to a desired network feature of the plurality of network features using a user input on the mobile station;

processing within the mobile station the selected original feature code to correspond to a local feature code for accessing the desired network feature; and

transmitting from the mobile station to the base station the local feature code corresponding to the desired network feature.

2. The method of claim 1, wherein the step of transmitting the local set of feature codes includes transmitting the local set of feature codes within a registration acknowledgment message.

3. The method of claim 1, wherein the step of transmitting the local set of feature codes includes transmitting the local set of feature codes within a Paging Channel Message.

4. The method of claim 1, wherein the step of transmitting the local set of feature codes includes transmitting the local set of feature codes within a Forward Traffic Channel Message.

5. The method of claim 4, wherein the Forward Traffic Channel Message is a Data Burst Message.

6. A method for accessing a plurality of network features enabled by a first set of feature codes using a mobile station having a second set of feature codes corresponding to the network features, the first set of feature codes having at least

one feature code differing from the second set of feature codes, the method comprising:

communicating between a network base station and the mobile station to register the mobile station with the network base station;

transmitting the first set of feature codes from the base station to the mobile station;

comparing the first set of feature codes with the second set of feature codes;

if the first set of feature codes is different from the second set of feature codes, storing the first set of feature codes within a memory of the mobile station;

selecting a second feature code from the second set of feature codes corresponding to a provisioned network feature of the plurality of network features using a user input on the mobile station;

processing within the mobile station the selected second feature code to correspond to a first feature code for accessing the provisioned network feature;

transmitting from the mobile station to the base station the first feature code corresponding to the provisioned network feature;

transmitting from the base station a message requesting acceptance of a surcharge for the provisioned network feature; and

awaiting acknowledgment of acceptance of the surcharge from the mobile station before transmitting a provisioned feature code corresponding to the provisioned feature code.

7. The method of claim 1, wherein the step of transmitting from the mobile station the local set of feature codes includes a request for acknowledgment of successful accessing of the desired network feature.

8. A method for remotely enabling a mobile station having an original set of feature codes to access at least one local set of feature codes from a base station for use in selecting one or more network features of a local network, wherein the original set of feature codes is at least partially incompatible with the local set of feature codes, the method comprising:

selecting an original feature code of the original set of feature codes corresponding to a desired network feature of the one or more network features using a user input on the mobile station;

transmitting the selected original feature code to the base station;

transmitting from the base station the local set of feature codes; correlating within the mobile station the selected original feature code to a corresponding local feature code from the local set of feature codes;

transmitting from the mobile station to the base station the corresponding local feature code for the desired network feature; and

transmitting from the base station an acknowledgment of successful selection of the desired network feature.

9. The method of claim 8, wherein the step of selecting an original feature code comprises entering a plurality of keystrokes at a keypad, the plurality of keystrokes including pressing a "Menu" key and selecting a pre-programmed menu location.

10. The method of claim 8, wherein the step of selecting an original feature code comprises entering a plurality of keystrokes at a keypad, the plurality of keystrokes including pressing a "star" key followed by at least one numeric key.

11. The method of claim 8, wherein the step of transmitting the local set of feature codes includes transmitting the local set of feature codes within a registration acknowledgment message.

12. The method of claim 8, wherein the step of transmitting the local set of feature codes includes transmitting the local set of feature codes within a Paging Channel Message.

13. The method of claim 8, wherein the step of transmitting the local set of feature codes includes transmitting the local set of feature codes within a Forward Traffic Channel Message.

14. The method of claim 13, wherein the Forward Traffic Channel Message is a Data Burst Message.

15. The method of claim 8, wherein one network feature of the one or more network features is a provisioned feature and further comprising the steps of transmitting from the base station a message requesting acceptance of a surcharge for the provisioned feature and awaiting acknowledgment of acceptance of the surcharge from the mobile station before transmitting a feature code corresponding to the provisioned feature.

16. A method for administering a wireless communications network comprising a base station and a plurality of mobile stations, each mobile station having an identity and a set of pre-determined selectable feature codes for accessing one or more network features, wherein a user of the mobile station is not required to initiate a procedure for changing the set of pre-determined selectable feature codes to a plurality of new feature codes, the method comprising: programming the mobile station with a feature code change option and at least one feature code change number for permitting over-the-air provisioning of network features;

when provisioning of a network feature is requested by the user of the mobile station, transmitting a Page Message from a base station to the mobile station, the Page Message including a request to verify the identity of the mobile phone and a first data field for the feature code change option;

receiving a Page Response Message from the mobile phone to the base station including the identity of the mobile phone;

transmitting a Channel Assignment Message from the base station to the mobile phone, the Channel Assignment Message instructing the mobile phone to set up a traffic channel;

transmitting a Data Burst Message from the base station to the mobile phone instructing the mobile phone to enter an over-the-air service provisioning process;

transmitting the at least one feature code change number corresponding to the requested network feature from the base station to the mobile phone;

transmitting from the mobile phone to the base station a response verifying the at least one feature code change number;

downloading a new feature code of said plurality of new feature codes for accessing the requested network feature from the base station to the mobile phone, the new feature code not included in the set of pre-determined selectable feature codes;

transmitting a message from the base station to the mobile phone for requesting activation of the requested network feature; and

acknowledging activation of the requested network feature.

17. The method of claim 16, further comprising the steps of transmitting from the base station a message requesting acceptance of a surcharge for the requested network feature and awaiting acknowledgment of acceptance of the sur-

charge from the mobile station before transmitting a feature code corresponding to the requested network feature.

18. A mobile phone having means for accessing a local set of feature codes for use in selecting at least one network feature of a local network having a base station, wherein the mobile phone has a pre-determined means for selecting an original set of feature codes, and wherein the original set of feature codes is at least partially incompatible with the local set of feature codes, the mobile phone comprising:

a transceiver for receiving a message transmitted from the base station, the message including the local set of feature codes, and for transmitting responses and requests to the base station;

a first memory means for storing the original set of feature codes;

a second memory means for storing the local set of feature codes;

a processor comprising;

an input for receiving the message from the transceiver;

a comparator means for comparing the local set of feature codes with the original set of feature codes; and

if the local set of feature codes is at least partially incompatible with the original set of feature codes, means for storing the local set of feature codes within the second memory means;

a correlation means for correlating a local feature code from the local set of feature codes to correspond an original feature code from the original set of feature codes in response to a user selection of the original feature code;

an output for sending the local feature code corresponding to the at least one network feature to the transceiver to transmit to the base station; and

a user input in communication with the processor for selecting the original feature code corresponding to the at least one network feature.

19. The mobile phone of claim 18, wherein the message is a registration acknowledgment message.

20. The mobile phone of claim 18, wherein the message is a Paging Channel Message.

21. The mobile phone of claim 18, wherein the message is a Forward Traffic Channel Message.

22. The mobile phone of claim 18, wherein the message is a Data Burst Message.

23. A mobile phone with a pre-determined means for selecting an original set of feature codes for activating one or more network features of a local network responsive to at least one of a local set of feature codes, the original set of feature codes having at least one original feature code that is incompatible with the local set of feature codes, the local network having a base station, the mobile phone comprising:

a user input in communication with a processor for selecting an original feature code of the original set of feature codes corresponding to a desired network feature of the one or more network features;

a transceiver for transmitting a request for the selected original feature code to the base station and for receiving from the base station the local set of feature codes; and

a memory means in communication with the processor for storing the local set of feature codes, the processor including means for correlating the selected original feature code to a corresponding local feature code from the local set of feature codes and causing the transceiver to transmit the corresponding local feature code for the desired network feature.

24. The mobile phone of claim 23, wherein the user input comprises a keypad including a "Menu" key and a plurality of function-selecting keys, wherein a combination of the "Menu" key followed by at least one function-selecting key selects an original feature code from a pre-programmed menu location in the memory means.

25. The mobile phone of claim 23, wherein the user input comprises a "star" key and a plurality of numeric keys, wherein a combination of the "star" key followed by at least one numeric key selects an original feature code.

26. A method for accessing a plurality of network features enabled by a first set of feature codes using a mobile station having a second set of feature codes corresponding to the network features, the first set of feature codes having at least one feature code differing from the second set of feature codes, the method comprising;

communicating between a network base station and the mobile station to register the mobile station with the network base station;

transmitting the first set of feature codes from the base station to the mobile station;

comparing the first set of feature codes with the second set of feature codes;

if the first set of feature codes is different from the second set of feature codes, storing the first set of feature codes within a memory of the mobile station;

selecting a second feature code from the second set of feature codes corresponding to a desired network feature of the plurality of network features using a user input on the mobile station; processing within the mobile station the selected second feature code to correspond to a first feature code for accessing the desired network feature;

transmitting from the mobile station to the base station the first feature code corresponding to the desired network feature; and

if the first set of feature codes includes a new feature code for a new network feature that is unavailable in the second set of feature codes of the mobile station, informing the user of the availability of the new feature code, and establishing means for the user to utilize the new feature code.

27. The method of claim 8, wherein the local set of feature codes includes an additional feature code for an additional network feature that does not correspond to an original feature code of the original set of feature codes, further comprising the steps of:

notifying the user of the additional feature code; and prompting the user for an acknowledgment of the additional feature code.

28. A mobile phone having means for accessing a new set of feature codes for use in selecting one or more network features of a local network having a base station, wherein the mobile phone has a pre-determined means for selecting an original set of feature codes, the mobile phone comprising:

a transceiver for receiving a message transmitted from the base station, the message including the new set of feature codes, and for transmitting responses and requests to the base station;

a first memory means for storing the original set of feature codes;

a second memory means for storing the new set of feature codes;

a processor comprising;

an input for receiving the message from the transceiver;

a comparator means for comparing the new set of feature codes with the original set of feature codes; and

if the new set of feature codes is different from the original set of feature codes, means for storing the new set of feature codes within the second memory means;

a correlation means for correlating a new feature code from the new set of feature codes to correspond an original feature code from the original set of feature codes in response to a user selection of the original feature code;

an output for sending the new feature code corresponding to the desired network feature to the transceiver to transmit to the base station;

a means for identifying an additional feature code from the new set of feature codes that does not correspond to an original feature code; and

notification means for notifying a user of the availability of the additional feature code; and

a user input in communication with the processor for selecting the original feature code corresponding to a desired network feature.

29. A mobile phone with a pre-determined means for selecting an original set of feature codes for activating one or more network features of a local network responsive to at

least one of a new set of feature codes, the local network having a base station, the mobile phone comprising:

a user input in communication with a processor for selecting an original feature code of the original set of feature codes corresponding to a desired network feature of the one or more network features;

a transceiver for transmitting a request for the selected original feature code to the base station and for receiving from the base station the new set of feature codes;

a memory means in communication with the processor for storing the new set of feature codes, the processor including means for correlating the selected original feature code to a corresponding new feature code from the new set of feature codes and causing the transceiver to transmit the corresponding new feature code for the desired network feature; and

wherein the new set of feature codes includes at least one local code that is unavailable in the original set of feature codes, the processor further including means to notify the user of the availability of the at least one local code.

* * * * *